Overall Test Result

Retry

0%

0% 0 Correct Answers

100% 369 Incorrect Answers

The practice test includes only multiple-choice questions (MCQs) due to development limitations. To achieve a good score, you need to review the premium PDF.

Correct Only Incorrect Only

Question: 1

Exam Heist

A large mobile network operating company is building a machine learning model to predict customers who are likely to unsubscribe from the service. The company plans to offer an incentive for these customers as the cost of churn is far greater than the cost of the incentive.
The model produces the following confusion matrix after evaluating on a test dataset of 100 customers:

n= 100	PREDICTED CHURN	PREDICTED CHURN
	Yes	No
ACTUAL Churn Yes	10	4
Actual No	10	76

Based on the model evaluation results, why is this a viable model for production?

- The model is 86% accurate and the cost incurred by the company as a result of false negatives is less than the false positives.
- The precision of the model is 86%, which is less than the accuracy of the model.
- The model is 86% accurate and the cost incurred by the company as a result of false positives is less than the false negatives. ✓
- The precision of the model is 86%, which is greater than the accuracy of the model.

Explanation:
There are more FP's than FN's, however the costs of FN's are far larger than that of FP's. So:numberof(FP) > numberof(FN), costperunit(FP) << costperunit(FN). This itself could suggest that totalcosts(FP) < totalcosts(FN), but would be somewhat subjective, since it is not stated how far the unitary costs are.What is suggested, however, is that the model is indeed viable (question asks WHY the model is viable, and not WHETHER it's viable). If the model didn't exist, there would be no way that there are FP's or FN's, but churns would still exist, which have the same cost as FN's.So it means the total costs with FP's must be less than the total costs with FN's (churns).

Question: 2

Exam Heist

A Machine Learning Specialist is designing a system for improving sales for a company. The objective is to use the large amount of information the company has on users' behavior and product preferences to predict which products users would like based on the users' similarity to other users. What should the Specialist do to meet this objective?

- Build a content-based filtering recommendation engine with Apache Spark ML on Amazon EMR
- Build a collaborative filtering recommendation engine with Apache Spark ML on Amazon EMR. ✓
- Build a model-based filtering recommendation engine with Apache Spark ML on Amazon EMR
- Build a combinative filtering recommendation engine with Apache Spark ML on Amazon EMR

Explanation:
The correct answer is B, building a collaborative filtering recommendation engine with Apache Spark ML on Amazon EMR.
Here's why:
The problem describes a scenario where predicting user preferences is based on the similarity of their behavior and preferences to other users. This aligns perfectly with the principles of collaborative filtering. Collaborative filtering algorithms leverage the collective wisdom of the user base to make recommendations. They find users who have similar tastes and preferences and then recommend items that those similar users have liked or purchased.
Option A, content-based filtering, focuses on the attributes of the items themselves. It recommends items similar to those a user has liked in the past, regardless of other users' preferences. This doesn't fit the problem's emphasis on user similarity.
Option C, model-based filtering, is a broader term that can encompass collaborative filtering, but the specific phrasing is less precise than "collaborative filtering." Furthermore, model-based approaches in collaborative filtering context still leverage user-item interaction data.
Option D, combinative filtering, is not a standard or well-defined term in recommendation systems. While recommendation systems can combine different approaches, it's not the core principle being applied here.
Apache Spark ML on Amazon EMR is an excellent choice for implementation. Spark ML provides scalable machine learning libraries well-suited for processing large datasets of user behavior and product preferences. Amazon EMR provides a managed Hadoop framework that simplifies the deployment and management of Spark clusters, making it easier to build and scale the recommendation engine. Given the potentially large dataset, using a distributed computing framework like Spark on EMR is highly beneficial.Therefore, building a collaborative filtering engine using Spark ML on EMR is the most appropriate and effective solution for the given objective.
Relevant links:

Collaborative Filtering: https://en.wikipedia.org/wiki/Collaborative_filtering
Apache Spark MLlib: https://spark.apache.org/mllib/
Amazon EMR: https://aws.amazon.com/emr/

Question: 3

Exam Heist

A Mobile Network Operator is building an analytics platform to analyze and optimize a company's operations using Amazon Athena and Amazon S3. The source systems send data in .CSV format in real time. The Data Engineering team wants to transform the data to the Apache Parquet format before storing it on Amazon S3.
Which solution takes the LEAST effort to implement?

- Ingest .CSV data using Apache Kafka Streams on Amazon EC2 instances and use Kafka Connect S3 to serialize data as Parquet
- Ingest .CSV data from Amazon Kinesis Data Streams and use Amazon Glue to convert data into Parquet. ✓
- Ingest .CSV data using Apache Spark Structured Streaming in an Amazon EMR cluster and use Apache Spark to convert data into Parquet.
- Ingest .CSV data from Amazon Kinesis Data Streams and use Amazon Kinesis Data Firehose to convert data into Parquet.

Explanation:

The question asks for the solution that requires the *least* effort to implement for transforming real-time CSV data into Parquet format and storing it on S3.

Here's why option B (Ingest .CSV data from Amazon Kinesis Data Streams and use Amazon Glue to convert data into Parquet) is the best choice:

Managed Services Minimization of Effort: AWS Glue is a fully managed extract, transform, and load (ETL) service. This drastically reduces the operational overhead compared to managing infrastructure like EC2 instances (option A) or EMR clusters (option C).

Kinesis Data Streams for Real-time Ingestion: Kinesis Data Streams is designed for real-time data ingestion, making it a suitable choice for handling the stream of CSV data.

Glue's Built-in Parquet Conversion: AWS Glue has built-in capabilities to read data from Kinesis Data Streams, transform it, and write it to S3 in Parquet format. This is typically done using Glue's DynamicFrames which provide schema evolution capabilities and make it easier to deal with the schema changes that may occur in CSV files. This eliminates the need for custom code to handle the conversion process itself.

Reduced Coding and Configuration: Using Glue involves less coding compared to options A and C. You mainly define the data source (Kinesis), the transformation logic using Glue's visual interface or Spark code, and the target location in S3.

Kinesis Firehose Limitations: While Kinesis Data Firehose (option D) can ingest data and write it to S3, its transformation capabilities are more limited than Glue's. Firehose supports basic transformations using Lambda functions, but for complex transformations like CSV-to-Parquet conversion, it's less efficient than using Glue's ETL capabilities. Option D would require a more complex Lambda function for the conversion. Also, Firehose does not natively support schema inference/handling the way Glue does with DynamicFrames which makes dealing with CSV schemas easier.

Options A and C require more effort due to managing infrastructure (EC2 and EMR), configuring Kafka Streams or Spark, and writing custom code for data transformation.

Therefore, option B provides the simplest and most managed approach to achieve the desired outcome with the least effort.

Supporting Links:

AWS Glue: https://aws.amazon.com/glue/
Amazon Kinesis Data Streams: https://aws.amazon.com/kinesis/data-streams/
Apache Parquet: https://parquet.apache.org/
AWS Glue DynamicFrames: https://docs.aws.amazon.com/glue/latest/dg/aws-glue-api-common.html

Question: 4

Exam Heist

A city wants to monitor its air quality to address the consequences of air pollution. A Machine Learning Specialist needs to forecast the air quality in parts per million of contaminates for the next 2 days in the city. As this is a prototype, only daily data from the last year is available.
Which model is MOST likely to provide the best results in Amazon SageMaker?

- Use the Amazon SageMaker k-Nearest-Neighbors (kNN) algorithm on the single time series consisting of the full year of data with a predictor_type of regressor.
- Use Amazon SageMaker Random Cut Forest (RCF) on the single time series consisting of the full year of data.
- Use the Amazon SageMaker Linear Learner algorithm on the single time series consisting of the full year of data with a predictor_type of regressor. ✓
- Use the Amazon SageMaker Linear Learner algorithm on the single time series consisting of the full year of data with a predictor_type of classifier.

Explanation:

Here's a detailed justification for why option C is the most likely to provide the best results, along with explanations of why the other options are less suitable in this context:

Why Option C (Amazon SageMaker Linear Learner with regressor) is the best choice:

Regression Task: The problem explicitly states the need to forecast air quality in parts per million (ppm), which is a continuous numerical value. This makes it a regression problem, where the goal is to predict a continuous output.

Linear Learner Suitability: Linear Learner is a good starting point for regression tasks, especially when dealing with a relatively small dataset (one year of daily data). It's computationally efficient and can often provide a reasonable baseline model.
Simplicity and Interpretability: With limited data, a more complex model might overfit. Linear Learner is simpler and less prone to overfitting, making it a practical choice for a prototype. Also, Linear Learner is more interpretable than other complex models, which helps debug and extract insights quickly.
Time Series Application: Although Linear Learner is not specifically designed for Time Series it can still be used with one-year data by carefully structuring the input features. For instance, the previous day's air quality can be one of the input features. With only one year data, it is difficult to perform time series decomposition, trend estimation, or seasonality modeling for more specialized methods.
Why Option A (Amazon SageMaker k-Nearest-Neighbors (kNN) with regressor) is less suitable:

Data Density Requirements: k-NN relies on finding similar data points in the feature space. With only a year of daily data, the density of data points might be insufficient for k-NN to make accurate predictions.
Curse of Dimensionality: If additional features are added (e.g., lagged air quality values), the feature space becomes higher-dimensional, which can degrade k-NN performance. This makes it more vulnerable when using only one year data.
Why Option B (Amazon SageMaker Random Cut Forest (RCF)) is unsuitable:

Anomaly Detection Purpose: RCF is primarily designed for anomaly detection, not forecasting. While it can identify unusual patterns in a time series, it's not the appropriate algorithm for predicting future air quality values.
Lack of Forecasting Capability: Even if anomalies are detected, RCF doesn't inherently provide a way to predict future values of the time series.
Why Option D (Amazon SageMaker Linear Learner with classifier) is incorrect:

Classification vs. Regression: As explained previously, forecasting air quality in ppm is a regression problem, not a classification problem. A classifier predicts discrete categories or classes, not continuous values. Therefore, Linear Learner is not the right choice.
Supporting Documentation:

Amazon SageMaker Linear Learner: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html
Amazon SageMaker k-NN: https://docs.aws.amazon.com/sagemaker/latest/dg/k-nearest-neighbors.html
Amazon SageMaker Random Cut Forest: https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html
In summary, given the limited data and the requirement for a quick prototype, using Amazon SageMaker Linear Learner with a regressor is the most appropriate and efficient approach.

Question: 5

Exam Heist

A Data Engineer needs to build a model using a dataset containing customer credit card information
How can the Data Engineer ensure the data remains encrypted and the credit card information is secure?

- Use a custom encryption algorithm to encrypt the data and store the data on an Amazon SageMaker instance in a VPC. Use the SageMaker DeepAR algorithm to randomize the credit card numbers.
- Use an IAM policy to encrypt the data on the Amazon S3 bucket and Amazon Kinesis to automatically discard credit card numbers and insert fake credit card numbers.
- Use an Amazon SageMaker launch configuration to encrypt the data once it is copied to the SageMaker instance in a VPC. Use the SageMaker principal component analysis (PCA) algorithm to reduce the length of the credit card numbers.
- Use AWS KMS to encrypt the data on Amazon S3 and Amazon SageMaker, and redact the credit card numbers from the customer data with AWS Glue. ✓

Explanation:
The correct answer is D because it provides a comprehensive approach to securing sensitive credit card data throughout the machine learning pipeline, addressing both encryption and redaction.
Here's a detailed justification:
Encryption at Rest and in Transit: Using AWS KMS (Key Management Service) to encrypt data on both Amazon S3 (where the raw data likely resides) and Amazon SageMaker (where the model is trained) ensures data is protected both when stored (at rest) and during transit (between services). KMS provides robust encryption using customer-managed keys or AWS-managed keys, giving the data engineer control over the encryption process. This aligns with security best practices and compliance requirements for handling sensitive data. https://aws.amazon.com/kms/
Data Redaction: Redacting the credit card numbers using AWS Glue is crucial. Encryption alone might not be sufficient if the model inadvertently learns patterns from the actual credit card numbers themselves. Redaction (removing or masking the sensitive information) minimizes the risk of exposing the real credit card data, even if the model is compromised. AWS Glue is a

serverless ETL (Extract, Transform, Load) service that can perform data transformation tasks like redaction efficiently. https://aws.amazon.com/glue/

Compliance and Security Best Practices: This combination of encryption and redaction aligns with common compliance standards like PCI DSS, which governs the handling of credit card data. It also follows the principle of least privilege by limiting the exposure of sensitive data to the model training process.

Why other options are incorrect:

A: While using a custom encryption algorithm and a VPC adds security, DeepAR is a time series forecasting algorithm and not suited for randomizing credit card numbers. Randomizing sensitive data directly compromises data integrity.
B: Using an IAM policy controls access to S3 but does not inherently encrypt the data. Automatically discarding and inserting fake credit card numbers corrupts the dataset and makes it unsuitable for model training.
C: Encrypting the data only when copied to the SageMaker instance leaves the data vulnerable when it is initially stored in S3. PCA is for dimensionality reduction, not secure data masking, and reducing the length isn't an appropriate anonymization technique. PCA could also inadvertently expose the data even if the credit card information is reduced in length.
In summary, only option D provides a robust and compliant solution by encrypting the data both at rest and in transit and redacting the sensitive credit card numbers using AWS Glue.

Question: 6

Exam Heist

A Machine Learning Specialist is using an Amazon SageMaker notebook instance in a private subnet of a corporate VPC. The ML Specialist has important data stored on the Amazon SageMaker notebook instance's Amazon EBS volume, and needs to take a snapshot of that EBS volume. However, the ML Specialist cannot find the Amazon SageMaker notebook instance's EBS volume or Amazon EC2 instance within the VPC.
Why is the ML Specialist not seeing the instance visible in the VPC?

- Amazon SageMaker notebook instances are based on the EC2 instances within the customer account, but they run outside of VPCs.
- Amazon SageMaker notebook instances are based on the Amazon ECS service within customer accounts.
- Amazon SageMaker notebook instances are based on EC2 instances running within AWS service accounts. ✓
- Amazon SageMaker notebook instances are based on AWS ECS instances running within AWS service accounts.

Explanation:
The correct answer is C: Amazon SageMaker notebook instances are based on EC2 instances running within AWS service accounts.
Here's why:
Amazon SageMaker notebook instances, while appearing to operate within a customer's specified VPC, are actually managed using EC2 instances within an AWS-managed service account. This means that the underlying EC2 instance hosting the notebook *doesn't* reside directly within the customer's VPC from an EC2 management perspective. The notebook instance is accessible within the customer's VPC through network interfaces created by SageMaker.
Option A is incorrect because SageMaker notebook instances operate within the customer's VPC, not outside of it. They are configured within the VPC, and network access controls are applied through VPC security groups.
Option B is incorrect because SageMaker notebook instances are not based on Amazon ECS. They are directly based on EC2 instances, but the EC2 instances exist in AWS managed service accounts, not customer accounts.
Option D is incorrect for the same reasons as B. They use EC2, not ECS. While SageMaker *can* use ECS for other features (like training jobs), the notebook instances are not directly running on ECS.
The ML Specialist is looking for an EC2 instance in *their* AWS account's VPC. The actual EC2 instance running the notebook is in an AWS service account's VPC. The network connection from the notebook to the internet or other VPC resources goes through an elastic network interface (ENI) that AWS creates in the customer's VPC. The notebook appears to be inside the VPC, but the underlying EC2 instance and EBS volume are not directly visible or manageable by the customer through standard EC2/EBS interfaces in their own AWS account. Therefore, the ML Specialist won't be able to directly locate the underlying EC2 instance or EBS volume through the EC2 or EBS consoles in their AWS account.
Further research:

Amazon SageMaker Notebook Instances - Security: This official AWS documentation clearly explains the networking and security aspects of SageMaker notebook instances within VPCs.
How Amazon SageMaker Secures Notebook Instance Traffic: A blog post further clarifying the traffic and networking related to SageMaker notebook instances.

Question: 7

Exam Heist

A Machine Learning Specialist is building a model that will perform time series forecasting using Amazon SageMaker. The Specialist has finished training the model and is now planning to perform load testing on the endpoint so they can configure Auto Scaling for the model variant.
Which approach will allow the Specialist to review the latency, memory utilization, and CPU utilization during the load test?

- Review SageMaker logs that have been written to Amazon S3 by leveraging Amazon Athena and Amazon QuickSight to visualize logs as they are being produced.
- Generate an Amazon CloudWatch dashboard to create a single view for the latency, memory utilization, and CPU utilization metrics that are outputted by Amazon SageMaker. ✓
- Build custom Amazon CloudWatch Logs and then leverage Amazon ES and Kibana to query and visualize the log data as it is generated by Amazon SageMaker.
- Send Amazon CloudWatch Logs that were generated by Amazon SageMaker to Amazon ES and use Kibana to query and visualize the log data.

Explanation:
The correct answer is B: Generate an Amazon CloudWatch dashboard to create a single view for the latency, memory utilization, and CPU utilization metrics that are outputted by Amazon SageMaker.
Here's a detailed justification:
Amazon SageMaker automatically emits several key performance metrics to Amazon CloudWatch, including latency, CPU utilization, and memory utilization. These metrics are crucial for monitoring endpoint health, performance, and resource consumption, especially during load testing for Auto Scaling configuration. CloudWatch provides a centralized platform for collecting and visualizing these metrics.
Option B directly leverages this built-in integration. By creating a CloudWatch dashboard, the Machine Learning Specialist can quickly visualize the latency, CPU utilization, and memory utilization metrics generated by SageMaker endpoints in real-time. This provides a consolidated view for monitoring the endpoint's behavior under increasing load, facilitating informed decisions about Auto Scaling configurations. CloudWatch allows for customized graphs, alerting based on metric thresholds, and historical data analysis.
Option A is less efficient and potentially more costly. While Athena and QuickSight can be used to analyze logs, it requires additional configuration to extract relevant performance metrics from the logs and may not provide the near real-time visibility needed during load testing. SageMaker already provides metrics directly to CloudWatch.
Option C involves custom CloudWatch Logs and Elasticsearch Service (Amazon ES) with Kibana. While powerful for log analysis, setting up custom logging and integrating with ES/Kibana is an unnecessary complication for monitoring basic resource utilization. SageMaker already sends these metrics to CloudWatch, making this approach redundant.
Option D, similar to option C, utilizes Amazon ES and Kibana but relies on SageMaker logs, which necessitates parsing through logs to extract the required performance metrics. This adds unnecessary complexity when CloudWatch directly provides these metrics.
In summary, CloudWatch provides a straightforward, integrated, and efficient method for monitoring SageMaker endpoint performance metrics such as latency, CPU utilization, and memory utilization. The dashboard provides the necessary real-time visibility during load testing, supporting effective Auto Scaling configuration.
Supporting Resources:

Monitor Amazon SageMaker with Amazon CloudWatch
Amazon CloudWatch Dashboards

Question: 8

Exam Heist

A manufacturing company has structured and unstructured data stored in an Amazon S3 bucket. A Machine Learning Specialist wants to use SQL to run queries on this data.
Which solution requires the LEAST effort to be able to query this data?

- Use AWS Data Pipeline to transform the data and Amazon RDS to run queries.
- Use AWS Glue to catalogue the data and Amazon Athena to run queries. ✓
- Use AWS Batch to run ETL on the data and Amazon Aurora to run the queries.
- Use AWS Lambda to transform the data and Amazon Kinesis Data Analytics to run queries.

Explanation:
The question asks for the solution that requires the least effort to query structured and unstructured data in an S3 bucket using SQL. Option B, using AWS Glue and Amazon Athena, is the most efficient approach. AWS Glue automatically discovers the schema of the data in S3 and creates a metadata catalog. This catalog allows Athena to then query the data directly using SQL without the need for complex ETL processes or data transformation. Athena is a serverless query service that uses standard SQL to analyze data stored in S3.

Options A, C, and D require more effort. Option A involves using AWS Data Pipeline to transform the data and then loading it into Amazon RDS. This involves creating a pipeline, which introduces more complexity. Option C involves using AWS Batch to run ETL jobs and then loading the data into Amazon Aurora. This involves setting up a Batch environment and creating ETL scripts. Option D involves using AWS Lambda to transform the data and then using Amazon Kinesis Data Analytics to run queries. While Kinesis Data Analytics can use SQL, it's more geared toward real-time streaming data. Using Lambda to transform the data would add considerable operational overhead.
Glue and Athena integrate seamlessly for querying data in S3. Athena directly works with the Glue Data Catalog. It avoids the complexities of data warehousing or complex transformations before querying. Thus making option B the least effort choice.
AWS Glue DocumentationAmazon Athena Documentation

Question: 9

Exam Heist

A Machine Learning Specialist is developing a custom video recommendation model for an application. The dataset used to train this model is very large with millions of data points and is hosted in an Amazon S3 bucket. The Specialist wants to avoid loading all of this data onto an Amazon SageMaker notebook instance because it would take hours to move and will exceed the attached 5 GB Amazon EBS volume on the notebook instance.
Which approach allows the Specialist to use all the data to train the model?

- Load a smaller subset of the data into the SageMaker notebook and train locally. Confirm that the training code is executing and the model parameters seem reasonable. Initiate a SageMaker training job using the full dataset from the S3 bucket using Pipe input mode. ✓
- Launch an Amazon EC2 instance with an AWS Deep Learning AMI and attach the S3 bucket to the instance. Train on a small amount of the data to verify the training code and hyperparameters. Go back to Amazon SageMaker and train using the full dataset
- Use AWS Glue to train a model using a small subset of the data to confirm that the data will be compatible with Amazon SageMaker. Initiate a SageMaker training job using the full dataset from the S3 bucket using Pipe input mode.
- Load a smaller subset of the data into the SageMaker notebook and train locally. Confirm that the training code is executing and the model parameters seem reasonable. Launch an Amazon EC2 instance with an AWS Deep Learning AMI and attach the S3 bucket to train the full dataset.

Explanation:
The best approach is **A. Load a smaller subset of the data into the SageMaker notebook and train locally. Confirm that the training code is executing and the model parameters seem reasonable. Initiate a SageMaker training job using the full dataset from the S3 bucket using Pipe input mode.**
Here's why:

SageMaker's Intended Use: SageMaker is designed for large-scale machine learning, and it's not efficient to manage large datasets on the notebook instance itself. The 5GB EBS volume confirms this.
Local Subset Testing: Training on a small subset on the notebook verifies the code's functionality, data loading, and initial hyperparameter settings without the delay of processing the entire dataset.
SageMaker Training Jobs: SageMaker Training Jobs are designed for large datasets. They handle data loading from S3 directly to the training instances efficiently.
Pipe Input Mode: Pipe mode is a streaming mechanism that allows SageMaker training instances to directly access data from S3 without requiring the entire dataset to be downloaded at once. This significantly reduces the memory footprint on the training instances and speeds up training. https://docs.aws.amazon.com/sagemaker/latest/dg/model-access-training-data.html
Alternatives are Less Efficient/Costly:

B: While using an EC2 instance with a Deep Learning AMI is an option, it's generally less efficient than using SageMaker Training Jobs, which are optimized for machine learning workloads and provide better integration with other AWS services. It also involves more manual configuration.
C: AWS Glue is primarily for ETL (Extract, Transform, Load) operations. While Glue can process data, using it solely for model training, even on a subset, is not its primary purpose. Glue also has associated costs which are generally higher than SageMaker.
D: Using an EC2 instance directly to train adds complexity to the deployment workflow that SageMaker training jobs can handle more efficiently.
In essence, the Specialist should leverage the power of SageMaker's training jobs and Pipe input mode to efficiently train the model using the full dataset stored in S3. This approach minimizes data transfer bottlenecks, uses SageMaker as it is intended, and avoids the need for large EBS volumes on the notebook instance.

Question: 10

Exam Heist

A Machine Learning Specialist has completed a proof of concept for a company using a small data sample, and now the Specialist is ready to implement an end-to-end solution in AWS using Amazon SageMaker. The historical training data is stored in Amazon RDS.
Which approach should the Specialist use for training a model using that data?

- Write a direct connection to the SQL database within the notebook and pull data in
- Push the data from Microsoft SQL Server to Amazon S3 using an AWS Data Pipeline and provide the S3 location within the notebook. ✓
- Move the data to Amazon DynamoDB and set up a connection to DynamoDB within the notebook to pull data in.
- Move the data to Amazon ElastiCache using AWS DMS and set up a connection within the notebook to pull data in for fast access.

Explanation:
Here's a detailed justification for why option B is the best approach, and why the other options are less suitable for training a model using Amazon SageMaker with data from Amazon RDS:
Justification for Option B (Push the data from Microsoft SQL Server to Amazon S3 using an AWS Data Pipeline and provide the S3 location within the notebook):
Scalability and SageMaker Compatibility: Amazon SageMaker is designed to work seamlessly with data stored in Amazon S3. S3 provides a highly scalable and durable storage solution suitable for large datasets commonly used in machine learning. SageMaker's training jobs can efficiently access and process data directly from S3.
Decoupling: Extracting data from RDS and storing it in S3 decouples the training process from the database. This is crucial because training can be resource-intensive and might impact the performance of the production RDS database if accessed directly.
AWS Data Pipeline Efficiency: AWS Data Pipeline automates the data transfer process from RDS to S3. It handles scheduling, error handling, and data transformations if needed, making the data preparation process more robust and manageable.
Data Versioning and Auditability: Storing data in S3 allows for easier versioning and auditing of the training data. This is important for reproducibility and compliance.
Cost-Effectiveness: S3 storage is generally more cost-effective than other storage options like DynamoDB or ElastiCache for large datasets used primarily for training.
Best Practice: Separating the database layer from the ML training workflow aligns with best practices for building scalable and maintainable machine learning pipelines.

Why other options are less suitable:

Option A (Direct Connection to SQL Database): While technically possible, directly connecting from a SageMaker notebook to the RDS database for training data poses several issues:

Performance Impact: Training can put a significant load on the database, potentially impacting other applications relying on the database.
Security Risks: Opening direct database access to a notebook increases the security risk.
Scalability Limitations: Limited by the database's performance and connection limits.
Option C (Move to DynamoDB): DynamoDB is a NoSQL database optimized for fast key-value lookups. While it's fast, it's not ideal for the large-scale data processing typically involved in machine learning training. Also, the cost can be significantly higher than S3 for storing large datasets.
Option D (Move to ElastiCache): ElastiCache is an in-memory caching service, designed for very fast data retrieval but is not meant to store large datasets. It's not suitable for storing the entire historical training data. Its primary use case is for caching frequently accessed data.
Authoritative Links:

Amazon S3: https://aws.amazon.com/s3/
Amazon SageMaker: https://aws.amazon.com/sagemaker/
AWS Data Pipeline: https://aws.amazon.com/datapipeline/
Amazon RDS: https://aws.amazon.com/rds/
In conclusion, moving the data to S3 using AWS Data Pipeline is the most scalable, cost-effective, and secure approach for training a model with SageMaker, while minimizing the impact on the production RDS database.

Question: 11

Exam Heist

A Machine Learning Specialist receives customer data for an online shopping website. The data includes demographics, past visits, and locality information. The
Specialist must develop a machine learning approach to identify the customer shopping patterns, preferences, and trends to enhance the website for better service and smart recommendations.
Which solution should the Specialist recommend?

- Latent Dirichlet Allocation (LDA) for the given collection of discrete data to identify patterns in the customer database.

- A neural network with a minimum of three layers and random initial weights to identify patterns in the customer database.
- Collaborative filtering based on user interactions and correlations to identify patterns in the customer database. ✓
- Random Cut Forest (RCF) over random subsamples to identify patterns in the customer database.

Explanation:
The best solution is collaborative filtering (Option C) because it directly addresses the problem of identifying customer shopping patterns, preferences, and trends based on user interactions. Collaborative filtering leverages the idea that users with similar tastes in the past will likely have similar tastes in the future. It analyzes user interactions (e.g., purchases, ratings, browsing history) and correlations between users or items to make recommendations. This aligns perfectly with the objective of enhancing the website with better service and smart recommendations.
Option A, Latent Dirichlet Allocation (LDA), is a topic modeling technique primarily used for discovering abstract "topics" in a collection of documents. While LDA could be used to find groups of products that are often purchased together, it doesn't directly model user preferences or enable personalized recommendations based on user similarities.
Option B, a neural network, is a powerful technique but requires substantial labeled data for training to accurately identify patterns. The prompt does not indicate the availability of labeled data to train the network. Developing a deep learning model without explicit training would also be overly complex given the requirements and other alternatives.
Option D, Random Cut Forest (RCF), is an anomaly detection algorithm. While it can identify unusual patterns in the data, it's not the most suitable approach for discovering general customer shopping preferences and providing recommendations. RCF is primarily for identifying outliers or fraudulent activities, not regular patterns.
Collaborative filtering, in contrast, is designed specifically for recommendation systems and pattern identification within user-item interaction data, making it the optimal choice. AWS provides services such as Amazon Personalize, which uses collaborative filtering (and other ML techniques) to create recommendation systems.
For further reading on Collaborative Filtering, refer to:

Wikipedia: https://en.wikipedia.org/wiki/Collaborative_filtering
Amazon Personalize Documentation: https://docs.aws.amazon.com/personalize/latest/dg/what-is-personalize.html

Question: 12

Exam Heist

A Machine Learning Specialist is working with a large company to leverage machine learning within its products. The company wants to group its customers into categories based on which customers will and will not churn within the next 6 months. The company has labeled the data available to the Specialist.
Which machine learning model type should the Specialist use to accomplish this task?

- Linear regression
- Classification ✓
- Clustering
- Reinforcement learning

Explanation:
The problem describes a scenario where the company aims to predict customer churn, a binary outcome (churn or not churn). This prediction task falls under the realm of supervised learning, specifically **classification**. Classification models learn from labeled data (customers labeled as churned or not churned) to assign new data points (customers) to predefined categories or classes.
Linear regression is suitable for predicting continuous numerical values, not for categorical outcomes like churn. Clustering, on the other hand, is an unsupervised learning technique used to group unlabeled data into clusters based on similarity. Since the problem specifies labeled data, clustering is not appropriate. Reinforcement learning involves training an agent to make decisions in an environment to maximize a reward. It's not applicable to predicting churn based on historical data.
Classification algorithms like logistic regression, decision trees, random forests, or gradient boosting machines are well-suited for this problem. These models can learn the patterns and relationships in the customer data to predict which customers are likely to churn. The labeled data provides the necessary target variable (churn or not churn) for training a supervised classification model.
In summary, because the task involves predicting a categorical outcome (churn or not churn) using labeled data, **classification** is the correct machine learning model type.
Here are some authoritative links for further research:

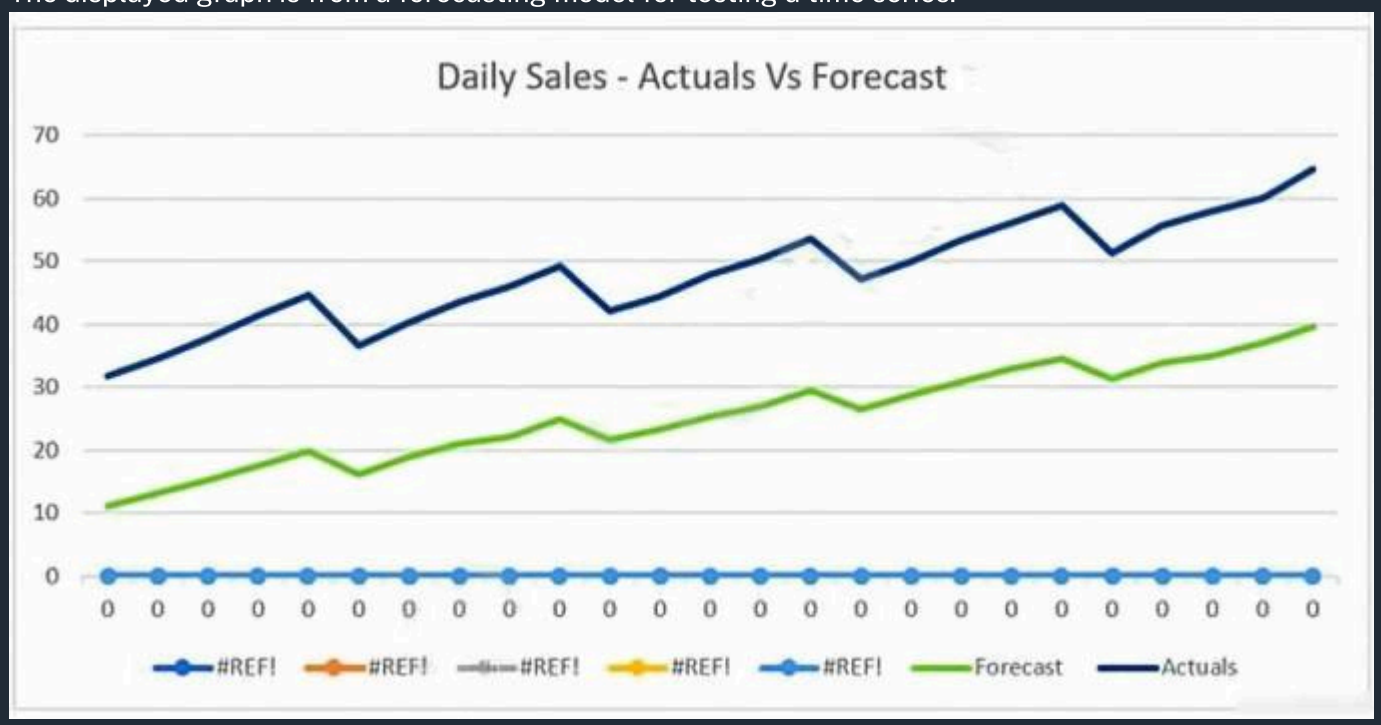
AWS Documentation on Machine Learning: https://aws.amazon.com/machine-learning/
Supervised Learning Concepts: https://developers.google.com/machine-learning/crash-course/classification/problem-framing

Classification Algorithms: https://scikit-learn.org/stable/tutorial/basic/tutorial.html

Question: 13

Exam Heist

The displayed graph is from a forecasting model for testing a time series.



Considering the graph only, which conclusion should a Machine Learning Specialist make about the behavior of the model?

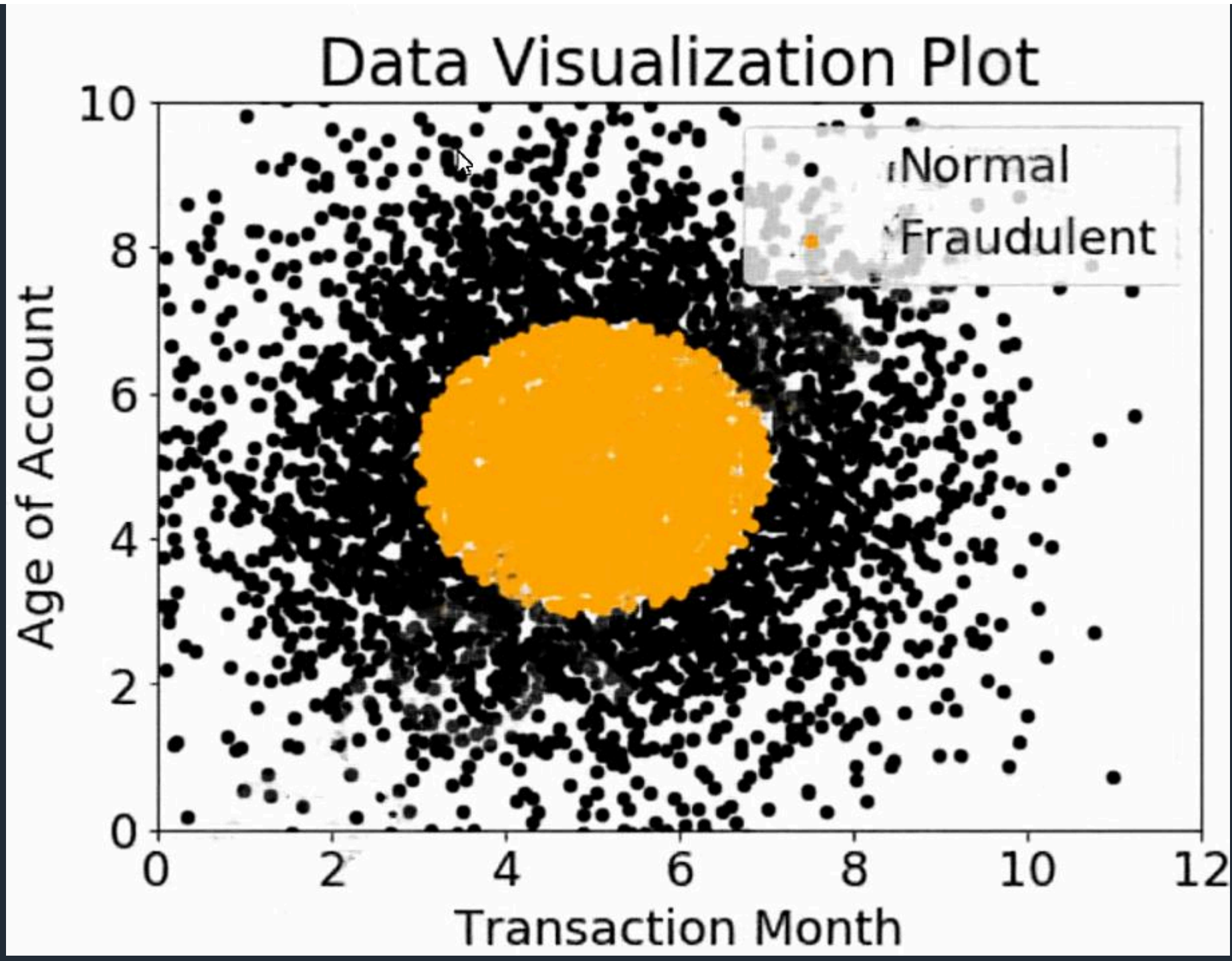
- The model predicts both the trend and the seasonality well ✓
- The model predicts the trend well, but not the seasonality.
- The model predicts the seasonality well, but not the trend.
- The model does not predict the trend or the seasonality well.

Explanation:
The model predicts both the trend and the seasonality well.
Reference:
https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

Question: 14

Exam Heist

A company wants to classify user behavior as either fraudulent or normal. Based on internal research, a Machine Learning Specialist would like to build a binary classifier based on two features: age of account and transaction month. The class distribution for these features is illustrated in the figure provided.



Based on this information, which model would have the HIGHEST accuracy?

- Long short-term memory (LSTM) model with scaled exponential linear unit (SELU)
- Logistic regression
- Support vector machine (SVM) with non-linear kernel ✓
- Single perceptron with tanh activation function

Explanation:
Answer is C. SVM sample use case is to put the dimensions into a higher hyperplane that can separates it. Seeing how separable it is, SVM can be used for it.

Question: 15

Exam Heist

A Machine Learning Specialist at a company sensitive to security is preparing a dataset for model training. The dataset is stored in Amazon S3 and contains
Personally Identifiable Information (PII).
The dataset:
⇨ Must be accessible from a VPC only.
⇨ Must not traverse the public internet.
How can these requirements be satisfied?

- Create a VPC endpoint and apply a bucket access policy that restricts access to the given VPC endpoint and the VPC. ✓
- Create a VPC endpoint and apply a bucket access policy that allows access from the given VPC endpoint and an Amazon EC2 instance.
- Create a VPC endpoint and use Network Access Control Lists (NACLs) to allow traffic between only the given VPC endpoint and an Amazon EC2 instance.
- Create a VPC endpoint and use security groups to restrict access to the given VPC endpoint and an Amazon EC2 instance

Explanation:
The correct answer is A: Create a VPC endpoint and apply a bucket access policy that restricts access to the given VPC endpoint and the VPC. This is the most secure and direct approach to fulfilling the requirements.

Here's a detailed justification:
VPC Endpoints for S3: VPC endpoints for S3 allow resources within your VPC to access S3 without traversing the public internet. This is critical for security-sensitive data, as it prevents PII from potentially being exposed to the public network.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
Eliminating Public Internet Access: By using a VPC endpoint, all traffic between the VPC and S3 remains within the Amazon network, avoiding the need for internet gateways or NAT instances.
Bucket Access Policies: S3 bucket access policies control who can access your S3 buckets and what actions they can perform. These policies are fundamental for securing S3 data.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html
Restricting Access by VPC Endpoint and VPC: The key to security is applying a bucket policy that *specifically* restricts access to only traffic originating from the created VPC endpoint and the VPC itself. This ensures that only authorized resources within that VPC can access the S3 bucket containing the PII. The policy would use the `aws:sourceVpc` and `aws:sourceVpcEndpoint` condition keys.
Why Other Options are Less Ideal:

B: While creating a VPC endpoint is correct, allowing access from an EC2 instance in addition to the VPC endpoint doesn't provide the same level of restriction. It's better to enforce access *only* via the endpoint. Also, implying the endpoint is only for the EC2 instance is incorrect, as the endpoint facilitates connection for the entire VPC.
C: NACLs are primarily for stateless traffic filtering at the subnet level. While NACLs could provide some degree of control, they are not the best tool for controlling access to S3 buckets. Bucket policies provide more granular and specific access control to the object level.
D: Security groups operate at the instance level, and while useful for securing EC2 instances, they are not suitable for directly controlling access to S3 buckets. Bucket Policies are meant to control S3 access.

In summary, creating a VPC endpoint provides a private connection to S3, and a bucket access policy that restricts access based on the VPC endpoint and VPC ID ensures that only authorized resources within that VPC can access the sensitive data, without ever exposing the data to the public internet. This approach offers the best combination of security and functionality for the given requirements.

Question: 16

Exam Heist

During mini-batch training of a neural network for a classification problem, a Data Scientist notices that training accuracy oscillates.
What is the MOST likely cause of this issue?

- ○ The class distribution in the dataset is imbalanced.
- ○ Dataset shuffling is disabled.
- ○ The batch size is too big.
- ○ The learning rate is very high. ✓

Explanation:
The most likely cause of oscillating training accuracy during mini-batch training is a very high learning rate (Option D). Here's why:
A high learning rate causes the optimization algorithm (e.g., gradient descent) to take large steps in the parameter space. Instead of converging smoothly towards the minimum of the loss function, these large steps can overshoot the minimum. This overshoot leads to the accuracy increasing and then decreasing as the model jumps back and forth across the minimum point, resulting in oscillations.
Imbalanced class distribution (Option A) typically leads to biased predictions favoring the majority class, impacting overall accuracy but not necessarily causing rapid oscillations. Disabled dataset shuffling (Option B) can lead to the network learning patterns specific to the order of data, making the training unstable or leading to suboptimal solutions but without the specific symptom of constant, rapid accuracy oscillation. A large batch size (Option C) tends to smooth out the gradient updates and reduces noise, so a batch size that is too big can slow down training or get trapped in local minima, but does not cause accuracy to oscillate.
The relationship between learning rate and convergence is fundamental to neural network training. When the learning rate is properly tuned, the model will converge steadily toward the minimum, improving accuracy with each iteration.A learning rate that is too small can also cause problems, namely slow convergence.When training with mini-batches, the learning rate must be carefully chosen, and it will generally differ depending on the batch size.
For more information on learning rates, consult resources like:

Machine Learning Mastery on Learning Rate: https://machinelearningmastery.com/learning-rate-for-deep-learning/

TensorFlow documentation on Optimizers: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Question: 17

Exam Heist

An employee found a video clip with audio on a company's social media feed. The language used in the video is Spanish. English is the employee's first language, and they do not understand Spanish. The employee wants to do a sentiment analysis.
What combination of services is the MOST efficient to accomplish the task?

- Amazon Transcribe, Amazon Translate, and Amazon Comprehend ✓
- Amazon Transcribe, Amazon Comprehend, and Amazon SageMaker seq2seq
- Amazon Transcribe, Amazon Translate, and Amazon SageMaker Neural Topic Model (NTM)
- Amazon Transcribe, Amazon Translate and Amazon SageMaker BlazingText

Explanation:
The most efficient solution to perform sentiment analysis on a Spanish audio clip for an English-speaking employee involves three AWS services working in sequence. Amazon Transcribe is used first to convert the Spanish audio from the video clip into Spanish text. https://aws.amazon.com/transcribe/
Next, Amazon Translate takes the transcribed Spanish text and translates it into English text, making it understandable to the employee. https://aws.amazon.com/translate/
Finally, Amazon Comprehend analyzes the translated English text and performs sentiment analysis to determine the emotional tone or attitude expressed in the content (e.g., positive, negative, neutral). https://aws.amazon.com/comprehend/
Option B is incorrect because while Amazon Transcribe is correct, Amazon Comprehend can directly perform sentiment analysis, rendering SageMaker seq2seq (a sequence-to-sequence model typically used for translation or text summarization, not directly sentiment analysis) unnecessary and less efficient.
Option C includes SageMaker Neural Topic Model (NTM), which is designed to discover topics within a body of text, not for sentiment analysis, making it an unsuitable choice.
Option D is incorrect because BlazingText is optimized for word embedding and text classification, not direct sentiment analysis, and requires considerable training and setup, making it less efficient than Amazon Comprehend for this pre-built task. The other services are correct.
Therefore, using Amazon Transcribe to convert audio to text, Amazon Translate to translate the text to English, and Amazon Comprehend to perform sentiment analysis is the most direct and efficient combination of services for this task as it uses readily available and optimized services.

Question: 18

Exam Heist

A Machine Learning Specialist is packaging a custom ResNet model into a Docker container so the company can leverage Amazon SageMaker for training. The
Specialist is using Amazon EC2 P3 instances to train the model and needs to properly configure the Docker container to leverage the NVIDIA GPUs.
What does the Specialist need to do?

- Bundle the NVIDIA drivers with the Docker image.
- Build the Docker container to be NVIDIA-Docker compatible. ✓
- Organize the Docker container's file structure to execute on GPU instances.
- Set the GPU flag in the Amazon SageMaker CreateTrainingJob request body.

Explanation:
The correct answer is B: Build the Docker container to be NVIDIA-Docker compatible. Here's why:
Leveraging GPUs within Docker containers for machine learning, especially on platforms like Amazon SageMaker using EC2 P3 instances, requires specific configurations to enable communication between the container and the host's NVIDIA GPUs. NVIDIA-Docker, now integrated into the NVIDIA Container Toolkit, is designed to solve this problem.
Option A is incorrect because directly bundling NVIDIA drivers into the Docker image isn't the recommended approach. This can lead to compatibility issues between the driver version in the container and the driver version on the host EC2 instance. Moreover, it significantly increases the image size. NVIDIA-Docker dynamically mounts the host's NVIDIA drivers into the container at runtime.
Option C is incorrect because the file structure within the Docker container is generally independent of whether GPUs are used. The application needs to be written to leverage GPUs (e.g., using CUDA or TensorFlow with GPU support), but the underlying container file structure doesn't inherently change.

Option D is incorrect because while the SageMaker `CreateTrainingJob` request defines the instance type (e.g., ml.p3.2xlarge) which determines whether GPUs are available, this alone is insufficient for the container to use the GPUs. The Docker container *must* be NVIDIA-Docker compatible to expose the GPUs to the application running within the container.
NVIDIA-Docker provides the necessary runtime components and tooling to bridge the gap between the container and the host's GPU resources. By ensuring the Docker image is NVIDIA-Docker compatible, the specialist guarantees that the CUDA libraries and drivers from the host EC2 instance are properly exposed to the container, enabling the ResNet model to effectively utilize the GPUs during training. This involves using the `nvidia-docker` command (or more recently, the NVIDIA Container Toolkit integration with Docker) during the container build or run process.
For authoritative information, refer to the NVIDIA Container Toolkit documentation: https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/index.htmland the AWS documentation on using Docker containers with SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/docker-container.html

Question: 19

Exam Heist

A Machine Learning Specialist is building a logistic regression model that will predict whether or not a person will order a pizza. The Specialist is trying to build the optimal model with an ideal classification threshold.
What model evaluation technique should the Specialist use to understand how different classification thresholds will impact the model's performance?

- ◯ Receiver operating characteristic (ROC) curve ✓
- ◯ Misclassification rate
- ◯ Root Mean Square Error (RMSE)
- ◯ L1 norm

Explanation:
Here's a detailed justification for why the correct answer is A (Receiver Operating Characteristic (ROC) curve):
The core problem revolves around understanding the impact of different classification thresholds on a logistic regression model's performance in predicting pizza orders. A classification threshold determines the point at which the model classifies a prediction as positive (will order pizza) versus negative (will not order pizza). Changing this threshold significantly affects the balance between true positives, true negatives, false positives, and false negatives.
An ROC curve is a graphical representation that plots the True Positive Rate (TPR, or sensitivity) against the False Positive Rate (FPR, or 1-specificity) at various threshold settings. By visualizing the ROC curve, the specialist can observe how the model's ability to correctly identify pizza orders (TPR) changes as the tolerance for incorrectly predicting pizza orders (FPR) varies. A curve that is closer to the top left corner of the graph indicates better performance, demonstrating higher TPR and lower FPR across thresholds. The area under the ROC curve (AUC) provides a single scalar value summarizing the overall performance.
Misclassification rate (option B) provides an overall error rate but doesn't reveal the trade-offs between different types of errors. Root Mean Square Error (RMSE) (option C) is used for regression problems and doesn't apply to classification threshold selection. L1 norm (option D) is a regularization technique used during model training to prevent overfitting, and it is not relevant for evaluating classification threshold performance.
Therefore, the ROC curve is the most appropriate evaluation technique because it helps the specialist visualize and analyze the performance of the logistic regression model across a range of classification thresholds, allowing for informed decisions on selecting the optimal threshold. This allows for the greatest balance between precision and recall based on the specifics of the business requirements.
Supporting resources:

ROC Curves and AUC: Understanding Evaluation Metrics for Imbalanced Classification
Receiver Operating Characteristic (ROC)

Question: 20

Exam Heist

An interactive online dictionary wants to add a widget that displays words used in similar contexts. A Machine Learning Specialist is asked to provide word features for the downstream nearest neighbor model powering the widget.
What should the Specialist do to meet these requirements?

- ◯ Create one-hot word encoding vectors.
- ◯ Produce a set of synonyms for every word using Amazon Mechanical Turk.
- ◯ Create word embedding vectors that store edit distance with every other word.
- ◯ Download word embeddings pre-trained on a large corpus. ✓

Explanation:

Here's a detailed justification for why option D is the best approach, along with supporting links:

The goal is to create word features that capture semantic similarity for a nearest neighbor model that recommends words used in similar contexts. Downloading pre-trained word embeddings is the most efficient and effective way to achieve this. Here's why:

Semantic Similarity: Word embeddings (like Word2Vec, GloVe, or fastText) are trained on massive text corpora to capture semantic relationships between words. Words used in similar contexts are placed closer to each other in the embedding space. This directly addresses the problem's requirement of finding words used in similar contexts.

Computational Efficiency: Training word embeddings from scratch can be extremely resource-intensive, requiring significant computational power and time. Downloading pre-trained embeddings leverages the work already done by others.

Generalization: Pre-trained embeddings have learned from a large corpus and generalize better to unseen data compared to training on a smaller, task-specific dataset.

Ready Availability: Many pre-trained word embedding models are readily available for download, making it a quick and easy solution.

Let's look at why the other options are less suitable:

A. One-hot encoding: One-hot encoding creates sparse, high-dimensional vectors that do not capture semantic relationships between words. Each word is treated as independent, and the model cannot understand that "king" and "queen" are more similar than "king" and "apple."

B. Synonyms using Mechanical Turk: While synonyms can be helpful, they only capture a narrow aspect of semantic similarity. Words used in similar contexts might not be perfect synonyms but still relevant to the task. Mechanical Turk also introduces potential inconsistencies and requires manual effort.

C. Edit distance: Edit distance measures the similarity between strings based on the number of edits required to transform one string into another. This is relevant for spelling correction but doesn't capture semantic meaning. "King" and "ring" would have a small edit distance but are not semantically related in the desired context.

In summary, downloading pre-trained word embeddings provides a quick, efficient, and effective way to capture semantic similarity for recommending words used in similar contexts. It leverages existing work and provides good generalization performance.

Supporting Links:

Word2Vec: https://papers.nips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f2c492369c905-Paper.pdf
GloVe: https://nlp.stanford.edu/projects/glove/
FastText: https://fasttext.cc/docs/en/crawl-vectors.html
Amazon SageMaker BlazingText (for training custom word embeddings, if needed): https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext.html

Question: 21

Exam Heist

A Machine Learning Specialist is configuring Amazon SageMaker so multiple Data Scientists can access notebooks, train models, and deploy endpoints. To ensure the best operational performance, the Specialist needs to be able to track how often the Scientists are deploying models, GPU and CPU utilization on the deployed SageMaker endpoints, and all errors that are generated when an endpoint is invoked.
Which services are integrated with Amazon SageMaker to track this information? (Choose two.)

- [x] AWS CloudTrail ✓
- [ ] AWS Health
- [ ] AWS Trusted Advisor
- [x] Amazon CloudWatch ✓
- [ ] AWS Config

Explanation:

The question requires identifying AWS services integrated with Amazon SageMaker for tracking model deployments, resource utilization, and endpoint invocation errors.

A. AWS CloudTrail: CloudTrail records API calls made to SageMaker. This includes actions like creating, updating, or deleting SageMaker resources (e.g., endpoints, models). Therefore, CloudTrail can track how often Data Scientists are deploying models because these deployment actions are logged as API calls.
[https://docs.aws.amazon.com/sagemaker/latest/dg/cloudtrail-logs.html]

D. Amazon CloudWatch: CloudWatch collects metrics and logs related to SageMaker resources. Specifically, CloudWatch can monitor CPU and GPU utilization on deployed SageMaker endpoints. It also captures logs generated during endpoint invocations, including any errors that occur. This allows for monitoring the health and performance of deployed models and troubleshooting issues. [https://docs.aws.amazon.com/sagemaker/latest/dg/monitoring-cloudwatch.html]
Why other options are incorrect:

B. AWS Health: AWS Health provides personalized information about the health of AWS services and resources. While it could potentially report on broader SageMaker service availability issues, it doesn't provide the granular endpoint-specific metrics and logs needed for detailed tracking.
C. AWS Trusted Advisor: Trusted Advisor provides best practice recommendations related to cost optimization, security, fault tolerance, and performance. It doesn't directly monitor endpoint utilization or track deployment frequency.
E. AWS Config: AWS Config tracks resource configurations and changes over time. While it might track the configuration of SageMaker endpoints, it does not directly provide metrics on resource utilization (CPU/GPU) or track error logs generated during endpoint invocations.
Therefore, AWS CloudTrail and Amazon CloudWatch provide the necessary capabilities for tracking model deployments, resource utilization, and endpoint invocation errors.

Question: 22

Exam Heist

A retail chain has been ingesting purchasing records from its network of 20,000 stores to Amazon S3 using Amazon Kinesis Data Firehose. To support training an improved machine learning model, training records will require new but simple transformations, and some attributes will be combined. The model needs to be retrained daily.
Given the large number of stores and the legacy data ingestion, which change will require the LEAST amount of development effort?

- Require that the stores to switch to capturing their data locally on AWS Storage Gateway for loading into Amazon S3, then use AWS Glue to do the transformation.
- Deploy an Amazon EMR cluster running Apache Spark with the transformation logic, and have the cluster run each day on the accumulating records in Amazon S3, outputting new/transformed records to Amazon S3.
- Spin up a fleet of Amazon EC2 instances with the transformation logic, have them transform the data records accumulating on Amazon S3, and output the transformed records to Amazon S3.
- Insert an Amazon Kinesis Data Analytics stream downstream of the Kinesis Data Firehose stream that transforms raw record attributes into simple transformed values using SQL. ✓

Explanation:
The best solution is to insert an Amazon Kinesis Data Analytics stream downstream of the Kinesis Data Firehose stream, transforming the data using SQL (Option D). This approach requires the least development effort due to several factors related to ease of integration, scalability, and cost-effectiveness.
Kinesis Data Analytics allows you to process streaming data using standard SQL queries. The transformation logic is simple to implement since SQL is used for data transformation, reducing the need for complex coding. It easily integrates with existing Kinesis Data Firehose setup. No significant changes are needed for the existing ingestion pipeline.
Kinesis Data Analytics is designed to automatically scale to handle the streaming data volume from 20,000 stores. Also, it has a pay-as-you-go pricing model, making it cost-effective because you only pay for the actual computation.
Alternatives are less attractive. Migrating stores to AWS Storage Gateway (Option A) involves significant changes to data capture infrastructure at each store and requires AWS Glue setup, leading to high development effort. Setting up EMR with Spark (Option B) is more complex and costly for simple transformations and requires managing a cluster. EC2 instances (Option C) require managing infrastructure and developing the transformation logic from scratch.
In summary, Kinesis Data Analytics provides a simple, scalable, and cost-effective solution for transforming data in real-time, minimizing development effort compared to alternatives.
Relevant Links:

Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/

Question: 23

Exam Heist

A Machine Learning Specialist is building a convolutional neural network (CNN) that will classify 10 types of animals. The Specialist has built a series of layers in a neural network that will take an input image of an animal, pass it through a series of convolutional and pooling layers, and then finally pass it through a dense and fully connected layer with 10 nodes. The Specialist would like to get an output from the neural network that is a probability distribution of how likely it is that the input image belongs to each of the 10 classes.
Which function will produce the desired output?

- Dropout
- Smooth L1 loss
- Softmax ✓
- Rectified linear units (ReLU)

Explanation:
The question asks for a function that outputs a probability distribution across 10 classes in a CNN.

Option C, Softmax, is the correct answer. Softmax is specifically designed to produce a probability distribution over multiple classes. It takes a vector of real numbers as input and transforms it into a probability distribution, where each element represents the probability of the input belonging to a specific class. The sum of these probabilities always equals 1. This aligns perfectly with the requirement of representing the likelihood of an input image belonging to each of the 10 animal classes.
Option A, Dropout, is a regularization technique used to prevent overfitting by randomly dropping out neurons during training. It doesn't produce a probability distribution.
Option B, Smooth L1 loss, is a loss function used to train regression models. It measures the difference between predicted and actual values and isn't relevant for generating a probability distribution for classification.
Option D, ReLU (Rectified Linear Unit), is an activation function that introduces non-linearity in a neural network. It outputs the input directly if it is positive, otherwise, it outputs zero. While ReLU is important in the hidden layers of a CNN, it doesn't produce a probability distribution for the output layer.
Therefore, only Softmax is designed to output a probability distribution, making it the appropriate function for the task.
Supporting Resources:

Softmax: https://en.wikipedia.org/wiki/Softmax_function
AWS Machine Learning Documentation: While AWS specific documentation doesn't usually focus on specific activation functions directly, searching for "classification layers AWS" can point towards solutions using softmax in their examples.

Question: 24

Exam Heist

A Machine Learning Specialist trained a regression model, but the first iteration needs optimizing. The Specialist needs to understand whether the model is more frequently overestimating or underestimating the target.
What option can the Specialist use to determine whether it is overestimating or underestimating the target value?

- Root Mean Square Error (RMSE)
- Residual plots ✓
- Area under the curve
- Confusion matrix

Explanation:
The correct answer is **B. Residual plots.** Here's why:
A residual plot is a graph that plots the residuals (the difference between the observed and predicted values) on the y-axis against the predicted or independent variable values on the x-axis. It provides a visual way to assess the spread and distribution of errors in a regression model. By examining the plot, a specialist can quickly determine if the model is systematically overestimating or underestimating the target variable.
If the points on the residual plot are randomly scattered around the horizontal axis (residual = 0), it suggests the model is making unbiased predictions. However, if there's a pattern, like a curved shape or more points above or below the line in certain regions, it indicates systematic error. For instance, if the majority of the points are above the x-axis for low predicted values and below for high predicted values, the model tends to underestimate low values and overestimate high values. Conversely, if the majority of the points are below the x-axis for low predicted values and above for high predicted values, the model tends to overestimate low values and underestimate high values.
RMSE (A) provides a single number representing the overall magnitude of errors but doesn't reveal the *direction* of those errors (overestimation vs. underestimation). AUC (C) is used primarily for classification models and isn't relevant to regression error analysis. A confusion matrix (D) is also a tool for classification problems, summarizing the performance by showing counts of true positives, true negatives, false positives, and false negatives.
Residual plots directly address the question by visually highlighting the nature of prediction errors. They are a critical diagnostic tool for understanding regression model bias.
Further Reading:

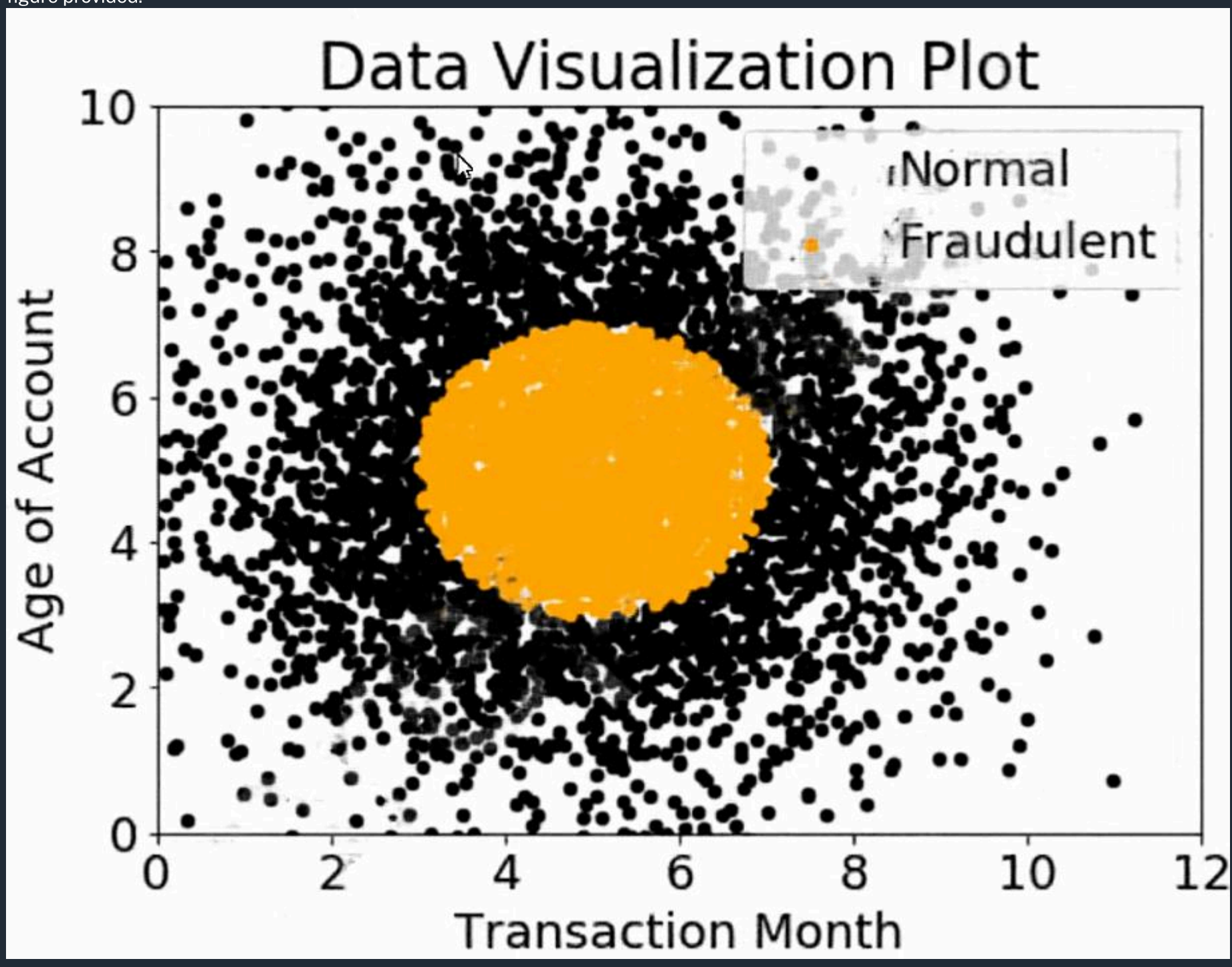
Residual Plots: https://www.statology.org/residual-plot/

Regression Diagnostics: https://online.stat.psu.edu/stat462/node/132/

Question: 25

Exam Heist

A company wants to classify user behavior as either fraudulent or normal. Based on internal research, a Machine Learning Specialist would like to build a binary classifier based on two features: age of account and transaction month. The class distribution for these features is illustrated in the figure provided.



Based on this information, which model would have the HIGHEST recall with respect to the fraudulent class?

- Decision tree
- Linear support vector machine (SVM)
- Naive Bayesian classifier ✓
- Single Perceptron with sigmoidal activation function

Explanation:

C. Naive Bayesian classifier.Naive Bayes classifiers are based on Bayes' theorem and are particularly useful for binary classification problems where the features are categorical or numerical. They are known to perform well on small datasets and are relatively fast to train and predict.In this case, there are two features, the age of the account and the transaction month, which can be used to classify user behavior as fraudulent or normal. Naive Bayes classifiers are suitable for this type of data as they work well with categorical features.

Question: 26

Exam Heist

A Machine Learning Specialist kicks off a hyperparameter tuning job for a tree-based ensemble model using Amazon SageMaker with Area Under the ROC Curve
(AUC) as the objective metric. This workflow will eventually be deployed in a pipeline that retrains and tunes hyperparameters each night to model click-through on data that goes stale every 24 hours.
With the goal of decreasing the amount of time it takes to train these models, and ultimately to decrease costs, the Specialist wants to reconfigure

the input hyperparameter range(s).
Which visualization will accomplish this?

- A histogram showing whether the most important input feature is Gaussian.
- A scatter plot with points colored by target variable that uses t-Distributed Stochastic Neighbor Embedding (t-SNE) to visualize the large number of input variables in an easier-to-read dimension.
- A scatter plot showing the performance of the objective metric over each training iteration.
- A scatter plot showing the correlation between maximum tree depth and the objective metric. ✓

Explanation:
The goal is to reduce training time and cost for a hyperparameter tuning job in SageMaker, specifically for a tree-based model using AUC as the objective metric, which is retrained daily. To optimize the hyperparameter range, the ML Specialist needs to understand how different hyperparameter values impact the objective metric (AUC).
Option A is incorrect because a histogram of input features, while useful for understanding data distribution, doesn't directly show the relationship between hyperparameters and the objective metric during the tuning process.
Option B is incorrect. While t-SNE can reduce dimensionality for visualization, it's primarily for understanding data structure and clustering, not hyperparameter optimization. It doesn't illustrate the direct impact of specific hyperparameters on the AUC.
Option C is incorrect because a scatter plot of the objective metric (AUC) over training iterations primarily shows the overall training progress and convergence. It does not directly reveal the correlation between *specific* hyperparameters and the AUC, which is critical for refining the hyperparameter ranges.
Option D, a scatter plot showing the correlation between maximum tree depth and the objective metric (AUC), is the most appropriate choice. Maximum tree depth is a key hyperparameter for tree-based models. By visualizing how different values of maximum tree depth affect the AUC score, the ML Specialist can identify optimal values. If, for example, the AUC plateaus after a certain tree depth, or even decreases due to overfitting, the Specialist can restrict the hyperparameter search range to lower depths. This reduces the search space and allows the tuning job to converge faster, leading to lower training costs and potentially improved model performance. This direct relationship helps in strategically adjusting the hyperparameter search space.
Here are authoritative links for further research:

Amazon SageMaker Hyperparameter Tuning: https://docs.aws.amazon.com/sagemaker/latest/dg/hyperparameter-tuning-how-it-works.html
Hyperparameter Tuning Strategies: https://aws.amazon.com/blogs/machine-learning/tuning-machine-learning-models-with-amazon-sagemaker/

Question: 27

Exam Heist

A Machine Learning Specialist is creating a new natural language processing application that processes a dataset comprised of 1 million sentences. The aim is to then run Word2Vec to generate embeddings of the sentences and enable different types of predictions.
Here is an example from the dataset:
"The quck BROWN FOX jumps over the lazy dog.`
Which of the following are the operations the Specialist needs to perform to correctly sanitize and prepare the data in a repeatable manner? (Choose three.)

- [ ] Perform part-of-speech tagging and keep the action verb and the nouns only.
- [x] Normalize all words by making the sentence lowercase. ✓
- [x] Remove stop words using an English stopword dictionary. ✓
- [ ] Correct the typography on "quck" to "quick.
- [ ] One-hot encode all words in the sentence.
- [x] Tokenize the sentence into words. ✓

Explanation:
The correct answer is B, C, and F. Here's why:

B. Normalize all words by making the sentence lowercase: Converting all text to lowercase ensures uniformity. The algorithm will treat "The" and "the" as the same word, reducing dimensionality and improving the model's ability to generalize. This is a standard text preprocessing step.

Text preprocessing - Machine Learning Crash Course

C. Remove stop words using an English stopword dictionary: Stop words (e.g., "the," "is," "a") are common words that don't carry significant meaning. Removing them reduces noise in the data, decreases the size of the vocabulary, and speeds up training. Libraries like NLTK provide standard stopword lists.

NLTK Stopwords Documentation

F. Tokenize the sentence into words: Tokenization is the process of breaking down the sentence into individual words (tokens). This is a fundamental step for Word2Vec and most NLP tasks because the algorithm operates on individual words.

Tokenization - SpaCy Documentation

Why the other options are incorrect:

A. Perform part-of-speech tagging and keep the action verb and the nouns only: While POS tagging can be useful in some NLP applications, it's not a necessary step for Word2Vec. Word2Vec focuses on the context of words within sentences, not specifically on their grammatical role. Furthermore, filtering only verbs and nouns might remove important contextual information carried by other word types.

D. Correct the typography on "quck" to "quick": While correcting spelling errors is generally beneficial, it's a form of data cleaning that can be addressed through custom scripts or rules as needed. It's not a universal prerequisite for Word2Vec and might not be necessary if the misspelled word is rare. The best option would be to train the model and then fix rare errors if the model performs poorly.

E. One-hot encode all words in the sentence: One-hot encoding represents words as sparse vectors, with a dimension for each word in the vocabulary. While one-hot encoding is useful in certain scenarios, Word2Vec generates dense word embeddings directly, making one-hot encoding redundant and computationally expensive. Word2Vec maps words to lower-dimensional vector spaces, capturing semantic relationships between words.

Question: 28

Exam Heist

A company is using Amazon Polly to translate plaintext documents to speech for automated company announcements. However, company acronyms are being mispronounced in the current documents.
How should a Machine Learning Specialist address this issue for future documents?

- ○ Convert current documents to SSML with pronunciation tags.
- ○ Create an appropriate pronunciation lexicon. ✓
- ○ Output speech marks to guide in pronunciation.
- ○ Use Amazon Lex to preprocess the text files for pronunciation

Explanation:

The correct answer is **B. Create an appropriate pronunciation lexicon.**

Here's why:

Amazon Polly offers the capability to use pronunciation lexicons to customize the pronunciation of specific words or phrases. This is particularly useful for acronyms, proper nouns, or domain-specific terms that Polly might mispronounce by default. A lexicon acts as a mapping between the written form of a word and its desired pronunciation, specified using either IPA (International Phonetic Alphabet) or X-SAMPA. When Polly processes text, it consults the lexicon and applies the defined pronunciations.

Option A, "Convert current documents to SSML with pronunciation tags," while also a valid approach, is more suitable for isolated cases or specific documents. SSML (Speech Synthesis Markup Language) allows fine-grained control over speech synthesis, including pronunciation through the `<phoneme>` tag. However, manually adding pronunciation tags to every instance of an acronym across many documents is cumbersome and not scalable. Creating a lexicon provides a centralized and reusable solution.

Option C, "Output speech marks to guide in pronunciation," is incorrect. Speech marks provide timing information and phonetic data about the synthesized speech, but they don't influence the pronunciation itself. They are more useful for synchronizing speech with other media, such as animations or subtitles.

Option D, "Use Amazon Lex to preprocess the text files for pronunciation," is also incorrect. Amazon Lex is a service for building conversational interfaces (chatbots) using speech and text. While Lex can understand spoken language, it's not designed for general text preprocessing for the purpose of improving pronunciation in Polly. Although Lex can utilize custom vocabulary for better speech recognition, this is not applicable in this scenario where the goal is to correct Polly's speech synthesis.

Therefore, creating a pronunciation lexicon is the most efficient and scalable way to address the mispronunciation of company acronyms in Amazon Polly. It allows for centralized management of pronunciation rules and ensures consistent pronunciation across all documents processed by Polly.

Supporting Documentation:

Amazon Polly Lexicons: https://docs.aws.amazon.com/polly/latest/dg/dg-managing-lexicons.html
Amazon Polly SSML: https://docs.aws.amazon.com/polly/latest/dg/supportedtags.html

Question: 29

Exam Heist

An insurance company is developing a new device for vehicles that uses a camera to observe drivers' behavior and alert them when they appear distracted. The company created approximately 10,000 training images in a controlled environment that a Machine Learning Specialist will use to train and evaluate machine learning models.
During the model evaluation, the Specialist notices that the training error rate diminishes faster as the number of epochs increases and the model is not accurately inferring on the unseen test images.
Which of the following should be used to resolve this issue? (Choose two.)

- [ ] Add vanishing gradient to the model.
- [x] Perform data augmentation on the training data. ✓
- [ ] Make the neural network architecture complex.
- [ ] Use gradient checking in the model.
- [x] Add L2 regularization to the model. ✓

Explanation:
The problem described indicates overfitting, where the model performs well on training data but poorly on unseen data. Options B and E directly address this.
Option B: Perform data augmentation on the training data. Data augmentation artificially increases the size of the training dataset by creating modified versions of existing images (e.g., rotations, flips, zooms). This exposes the model to a wider variety of examples and helps it generalize better, reducing overfitting. By augmenting the original 10,000 images, the model becomes more robust to variations in driver behavior captured by the camera, improving its performance on unseen test images.
Option E: Add L2 regularization to the model. L2 regularization adds a penalty term to the loss function that is proportional to the square of the magnitude of the weights. This encourages the model to learn smaller weights, which simplifies the model and reduces its sensitivity to noise in the training data, therefore mitigating overfitting. It penalizes large weights, preventing the model from memorizing the training data.
Why other options are less suitable:

A. Add vanishing gradient to the model. Vanishing gradients are the opposite problem (model not learning effectively), not overfitting.
C. Make the neural network architecture complex. Increasing the complexity of the model would likely exacerbate the overfitting problem. A more complex model has more parameters and is more capable of memorizing the training data.
D. Use gradient checking in the model. Gradient checking verifies the correctness of the backpropagation algorithm but does not directly address overfitting.
In summary, data augmentation increases the diversity of the training data, while L2 regularization simplifies the model, both contributing to better generalization and reducing overfitting.
Supporting Links:

Data Augmentation: https://www.tensorflow.org/tutorials/images/data_augmentation
Regularization: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization

Question: 30

Exam Heist

When submitting Amazon SageMaker training jobs using one of the built-in algorithms, which common parameters MUST be specified? (Choose three.)

- [ ] The training channel identifying the location of training data on an Amazon S3 bucket.
- [ ] The validation channel identifying the location of validation data on an Amazon S3 bucket.
- [x] The IAM role that Amazon SageMaker can assume to perform tasks on behalf of the users. ✓
- [ ] Hyperparameters in a JSON array as documented for the algorithm used.
- [x] The Amazon EC2 instance class specifying whether training will be run using CPU or GPU. ✓
- [x] The output path specifying where on an Amazon S3 bucket the trained model will persist. ✓

Explanation:

The correct answer is CEF because these parameters are essential for SageMaker to successfully train a model using built-in algorithms. Let's break down why each choice is crucial.

C. The IAM role: SageMaker needs permissions to access resources on your behalf, such as reading training data from S3, writing the trained model back to S3, and potentially accessing other AWS services. The IAM role grants SageMaker these necessary permissions. Without the correct IAM role, the training job will fail due to authorization errors. IAM Role documentation explains how to create and manage IAM roles for SageMaker.

E. The Amazon EC2 instance class: This parameter specifies the type of compute instance (CPU or GPU) that SageMaker will use for training. This selection significantly impacts performance and cost. Choosing a suitable instance depends on the algorithm and data size; GPU instances are often preferred for deep learning models due to their parallel processing capabilities. Amazon EC2 Instance Types provides an overview of available instances and their specifications.

F. The output path: This parameter tells SageMaker where to save the trained model (the model artifacts) in an S3 bucket after the training job completes. If this isn't specified, the model won't be persisted, and you won't be able to deploy it for inference. It acts as the final destination for your training process.

Now let's look at why the other options are less critical or not always required:

A. The training channel: While a training channel (pointing to training data in S3) is fundamental for model training, built-in algorithms require specific channel names like 'train' for training data, which SageMaker expects. This differs from custom training scripts where you explicitly define how channels are named and used.

B. The validation channel: A validation channel is optional. Although using a validation dataset is highly recommended for monitoring model performance during training and preventing overfitting, SageMaker can function and train a model even without a validation channel, using the training data alone.

D. Hyperparameters in a JSON array: Hyperparameters are often algorithm-specific and influence how the model learns. However, SageMaker built-in algorithms typically offer default values for most hyperparameters. So, while tuning them can improve model performance, explicitly specifying them isn't *always* mandatory for the training job to run. You can choose to use the algorithm defaults.

Question: 31

Exam Heist

A monitoring service generates 1 TB of scale metrics record data every minute. A Research team performs queries on this data using Amazon Athena. The queries run slowly due to the large volume of data, and the team requires better performance.
How should the records be stored in Amazon S3 to improve query performance?

- CSV files
- Parquet files ✓
- Compressed JSON
- RecordIO

Explanation:

Here's a detailed justification for why Parquet files (Option B) are the best choice for improving query performance in this scenario, along with explanations and relevant links:

The core problem is that the Research team's queries are slow due to the sheer volume of data (1 TB/minute) when using Amazon Athena. Athena's performance hinges on efficient data scanning. The goal is to minimize the amount of data Athena needs to read to answer the queries.

Parquet is a columnar storage format. Unlike row-based formats like CSV or JSON, Parquet stores data by columns. This is crucial because analytical queries (like those run by the Research team) often only need to access a subset of columns. With Parquet, Athena can read only the specific columns required for the query, dramatically reducing I/O and processing time. This principle is known as "columnar projection."

CSV files (Option A) and JSON files (Option C) are row-based formats. Athena would have to scan entire rows even if the query only needs one or two columns, making them inefficient for large datasets. Compressing JSON (Option C) helps reduce storage space but doesn't address the fundamental problem of inefficient scanning because it's still a row-based format.

RecordIO (Option D) is a format designed primarily for sequential reading of records, which is common in machine learning training pipelines. While it can be used for storage, it's not optimized for the kind of ad-hoc analytical queries that Athena is intended for. It also doesn't offer the column projection benefits of Parquet.

Furthermore, Parquet supports efficient data compression and encoding schemes, further reducing storage space and improving query performance. Athena can leverage these optimizations natively. Parquet files are also self-describing, meaning they include metadata about the schema, allowing Athena to understand the data structure without requiring external schema definitions in some cases.

In summary, Parquet's columnar storage, efficient compression, and ability to facilitate columnar projection make it the ideal choice for improving Athena query performance on large-scale datasets of metrics. Other formats don't offer the same level of optimization for analytical workloads.

Authoritative Links:

Apache Parquet: https://parquet.apache.org/
Amazon Athena Best Practices: https://docs.aws.amazon.com/athena/latest/ug/best-practices.html (search for "columnar storage" or "Parquet")
AWS Big Data Blog - Top 10 Performance Tuning Tips for Amazon Athena: https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/ (Tip #1 is about using columnar formats)

Question: 32

Exam Heist

Machine Learning Specialist is working with a media company to perform classification on popular articles from the company's website. The company is using random forests to classify how popular an article will be before it is published. A sample of the data being used is below.

Article_Title	Author	Top_Keywords	Day_Of_Week	URL_of_Article	Page_Views
Building a Big Data Platform	Jane Doe	Big Data, Spark, Hadoop	Tuesday	http://examplecorp.com/data_platform.html	1300456
Getting Started with Deep Learning	John Doe	Deep Learning, Machine Learning, Spark	Tuesday	http://examplecorp.com/started_deep_learning.html	1230661
MXNet ML Guide	Jane Doe	Machine Learning, MXNet, Logistic Regression	Thursday	http://examplecorp.com/mxnet_guide.html	937291
Intro to NoSQL Databases	Mary Major	NoSQL, Operations, Database	Monday	http://examplecorp.com/nosql_intro_guide.html	407812

Given the dataset, the Specialist wants to convert the Day_Of_Week column to binary values.
What technique should be used to convert this column to binary values?

- ○ Binarization
- ○ One-hot encoding ✓
- ○ Tokenization
- ○ Normalization transformation

Explanation:

B. One-hot encoding.

One-hot encoding is used to convert categorical variables into binary values by creating separate columns for each category. In this case, each day of the week would be represented as a separate binary column, where a value of 1 indicates the presence of that day and 0 indicates its absence.

Question: 33

Exam Heist

A gaming company has launched an online game where people can start playing for free, but they need to pay if they choose to use certain features. The company needs to build an automated system to predict whether or not a new user will become a paid user within 1 year. The company has gathered a labeled dataset from 1 million users.
The training dataset consists of 1,000 positive samples (from users who ended up paying within 1 year) and 999,000 negative samples (from users who did not use any paid features). Each data sample consists of 200 features including user age, device, location, and play patterns.
Using this dataset for training, the Data Science team trained a random forest model that converged with over 99% accuracy on the training set. However, the prediction results on a test dataset were not satisfactory
Which of the following approaches should the Data Science team take to mitigate this issue? (Choose two.)

- ☐ Add more deep trees to the random forest to enable the model to learn more features.
- ☐ Include a copy of the samples in the test dataset in the training dataset.
- ☐ Generate more positive samples by duplicating the positive samples and adding a small amount of noise to the duplicated data. ✓
- ☐ Change the cost function so that false negatives have a higher impact on the cost value than false positives. ✓

- [ ] Change the cost function so that false positives have a higher impact on the cost value than false negatives.

Explanation:
The provided scenario describes a classic case of class imbalance, where the number of negative samples (999,000) significantly outweighs the number of positive samples (1,000). The 99% accuracy on the training set is misleading because the model is likely predicting nearly everything as negative, achieving high accuracy simply by correctly classifying the vast majority of negative examples. This is a critical problem, particularly for the gaming company, as it wants to correctly identify users likely to convert to paying customers (positive samples).
Let's justify why options C and D are the correct choices:

C. Generate more positive samples by duplicating the positive samples and adding a small amount of noise to the duplicated data. This is an oversampling technique, specifically a form of data augmentation. By creating synthetic positive samples, the model is exposed to a wider range of variations within the positive class, helping it to learn the characteristics of positive users more effectively. Adding noise helps to prevent overfitting to the original positive samples. This addresses the core issue of class imbalance by increasing the representation of the minority class. It helps the model to better distinguish between positive and negative samples and improve the generalization to unseen data. More information on oversampling techniques can be found here: https://developers.google.com/machine-learning/data-prep/transform/resample
D. Change the cost function so that false negatives have a higher impact on the cost value than false positives. Because the business goal focuses on identifying potential paying customers, misclassifying a paying customer as a non-paying one (false negative) is more costly than misclassifying a non-paying user as a paying one (false positive). By assigning a higher cost to false negatives in the cost function (e.g., using a weighted loss function), the model will be penalized more severely for failing to identify potential paying customers. This adjustment encourages the model to prioritize recall (identifying all positive cases) over precision (avoiding false positives), which is appropriate given the business context. This technique is explained in detail in https://www.tensorflow.org/tutorials/structured_data/imbalanced_data.
Now let's analyze why the other options are incorrect:

A. Add more deep trees to the random forest to enable the model to learn more features. While increasing the complexity of the model *might* help in some cases, it is more likely to exacerbate overfitting to the majority class, especially in the presence of severe class imbalance. The fundamental issue is not the lack of learning capacity but the skewed distribution of classes.
B. Include a copy of the samples in the test dataset in the training dataset. This is a severe mistake that leads to data leakage. The model will be unfairly trained on data it will later be tested on, resulting in an artificially inflated and unrealistic performance evaluation.
E. Change the cost function so that false positives have a higher impact on the cost value than false negatives. This would worsen the situation. False positives are less detrimental to the gaming company, so increasing their cost is counterproductive.

Question: 34

Exam Heist

A Data Scientist is developing a machine learning model to predict future patient outcomes based on information collected about each patient and their treatment plans. The model should output a continuous value as its prediction. The data available includes labeled outcomes for a set of 4,000 patients. The study was conducted on a group of individuals over the age of 65 who have a particular disease that is known to worsen with age. Initial models have performed poorly. While reviewing the underlying data, the Data Scientist notices that, out of 4,000 patient observations, there are 450 where the patient age has been input as 0. The other features for these observations appear normal compared to the rest of the sample population
How should the Data Scientist correct this issue?

- [ ] Drop all records from the dataset where age has been set to 0.
- [ ] Replace the age field value for records with a value of 0 with the mean or median value from the dataset
- [ ] Drop the age feature from the dataset and train the model using the rest of the features.
- [ ] Use k-means clustering to handle missing features ✓

Explanation:
The correct approach to handle the 450 instances of age being 0 is to use a more sophisticated imputation technique, specifically k-means clustering, as option D suggests. While options A and C involve data loss which should generally be avoided unless absolutely necessary, and option B uses a simplistic imputation which can lead to biases.
Here's why k-means clustering is a better choice in this scenario:
Preserves Data: Unlike dropping the records (A) or the entire age feature (C), k-means clustering maintains all data points for training. This is crucial, especially given the relatively small dataset size of 4,000.
Contextual Imputation: k-means clustering leverages the relationships between *all* features to impute the missing age values. It groups similar patients based on their other features, and then uses the average age within the cluster to estimate the missing values. This is more informed than simply using the overall mean or median (B).

Reduces Bias: Replacing missing values with the mean or median introduces bias. This is because it does not consider the specific characteristics of the patient whose age is missing. k-means clustering helps to mitigate this by imputing values based on similar patient profiles.
Captures Non-Linear Relationships: k-means, being a clustering algorithm, can implicitly capture non-linear relationships between features, potentially leading to better imputation accuracy.
Domain Relevance: Given that the disease worsens with age, age is a crucial feature for the model. Therefore, it is preferable to impute missing values accurately rather than discard the feature entirely.

Why the other options are less suitable:

A (Dropping Records): Removing 450 records (over 10% of the data) is significant data loss, particularly problematic with a dataset of only 4,000. This can severely impact the model's accuracy and generalizability.
B (Mean/Median Imputation): Replacing missing age values with the mean or median age ignores the relationships between age and other features. This can introduce significant bias and lead to inaccurate predictions.
C (Dropping the Age Feature): Given the domain knowledge that the disease worsens with age, dropping the age feature would be a detrimental loss of valuable information that could improve model performance.
In summary, using k-means clustering for imputation is the most appropriate choice because it preserves data, considers feature relationships, reduces bias, and retains a critical feature for the model. This results in a more accurate and reliable model for predicting patient outcomes.
Authoritative Links:

Handling Missing Data: https://developers.google.com/machine-learning/data-prep/transform/missing-values
Imputation Techniques: https://scikit-learn.org/stable/modules/impute.html
k-Means Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Question: 35

Exam Heist

A Data Science team is designing a dataset repository where it will store a large amount of training data commonly used in its machine learning models. As Data
Scientists may create an arbitrary number of new datasets every day, the solution has to scale automatically and be cost-effective. Also, it must be possible to explore the data using SQL.
Which storage scheme is MOST adapted to this scenario?

- Store datasets as files in Amazon S3. ✓
- Store datasets as files in an Amazon EBS volume attached to an Amazon EC2 instance.
- Store datasets as tables in a multi-node Amazon Redshift cluster.
- Store datasets as global tables in Amazon DynamoDB.

Explanation:
The best storage scheme for the Data Science team's dataset repository is **A. Store datasets as files in Amazon S3.** Here's why:

Scalability: Amazon S3 is designed for virtually unlimited storage capacity. It can automatically scale to accommodate the daily creation of new datasets without manual intervention. https://aws.amazon.com/s3/
Cost-Effectiveness: S3 offers various storage classes (Standard, Intelligent-Tiering, Standard-IA, Glacier) optimized for different access patterns and cost requirements. This allows the team to store less frequently accessed datasets in cheaper tiers, optimizing overall costs.
SQL Exploration: While S3 itself isn't a SQL database, services like Amazon Athena and Amazon Redshift Spectrum enable querying data directly in S3 using SQL. This allows Data Scientists to explore the data without moving it to a separate database. https://aws.amazon.com/athena/ , https://aws.amazon.com/redshift/spectrum/
Data Lake Foundation: S3 is commonly used as the foundation for data lakes due to its scalability, cost-effectiveness, and integration with other AWS services. It supports storing various data formats suitable for machine learning models.
Alternative B (EBS): EBS volumes are attached to specific EC2 instances and do not offer the same level of automatic scalability or cost-effectiveness as S3. Managing storage across multiple EBS volumes can become complex.
Alternative C (Redshift): Redshift is primarily a data warehouse for structured data. While it can store datasets, it's less suitable for storing arbitrary files and might be more expensive for this purpose compared to S3, especially given the potentially high volume of new datasets. Redshift is optimized for analytical queries, but for simple exploration using SQL, Athena over S3 is more appropriate.
Alternative D (DynamoDB): DynamoDB is a NoSQL database designed for fast key-value lookups. It's not suitable for storing large datasets or performing complex SQL queries. Also, storing entire datasets in DynamoDB would be very expensive.

Therefore, S3 offers the best combination of scalability, cost-effectiveness, and SQL exploration capabilities for the Data Science team's needs.

Question: 36

Exam Heist

A Machine Learning Specialist deployed a model that provides product recommendations on a company's website. Initially, the model was performing very well and resulted in customers buying more products on average. However, within the past few months, the Specialist has noticed that the effect of product recommendations has diminished and customers are starting to return to their original habits of spending less. The Specialist is unsure of what happened, as the model has not changed from its initial deployment over a year ago.
Which method should the Specialist try to improve model performance?

- The model needs to be completely re-engineered because it is unable to handle product inventory changes.
- The model's hyperparameters should be periodically updated to prevent drift.
- The model should be periodically retrained from scratch using the original data while adding a regularization term to handle product inventory changes
- The model should be periodically retrained using the original training data plus new data as product inventory changes. ✓

Explanation:

The correct answer is D because the scenario describes a classic case of model drift. Model drift occurs when the statistical properties of the target variable, or the input features, change over time, leading to a decline in the model's predictive performance. In this case, customer behavior related to product recommendations has changed since the model was initially trained. This could be due to new trends, new products, competitor strategies, or a change in the customer demographic.
Option A is incorrect because re-engineering the entire model is often overkill for addressing model drift. Retraining is usually the first step.
Option B suggests updating hyperparameters, but this only addresses the model's internal settings and not the fundamental changes in the underlying data distribution. Hyperparameter tuning addresses model optimization with the same input data distribution but will not solve the drift problem.
Option C suggests retraining with original data and adding a regularization term. While regularization helps prevent overfitting, it won't address the core issue of the model not being exposed to the new data patterns reflecting recent customer behavior and product inventory changes. The original data is stale and no longer represents the current reality.
Retraining the model periodically with new data (Option D) allows the model to learn the current patterns and trends in customer behavior. This ensures that the model's predictions remain relevant and accurate over time, combating the effects of model drift. This is a common and effective strategy in machine learning model maintenance. By incorporating new data that reflects current product inventory changes and customer preferences, the model is better equipped to adapt to evolving market conditions and maintain its performance.
For further information, consider researching the following topics:

Model Drift: https://www.evidentlyai.com/blog/model-drift-detection
Retraining strategies: AWS documentation on model retraining.
Concept Drift: https://towardsdatascience.com/handling-concept-drift-in-machine-learning-65ff54a15b16

Question: 37

Exam Heist

A Machine Learning Specialist working for an online fashion company wants to build a data ingestion solution for the company's Amazon S3-based data lake.
The Specialist wants to create a set of ingestion mechanisms that will enable future capabilities comprised of:
- Real-time analytics
- Interactive analytics of historical data
- Clickstream analytics
- Product recommendations

Which services should the Specialist use?

- AWS Glue as the data catalog; Amazon Kinesis Data Streams and Amazon Kinesis Data Analytics for real-time data insights; Amazon Kinesis Data Firehose for delivery to Amazon ES for clickstream analytics; Amazon EMR to generate personalized product recommendations ✓
- Amazon Athena as the data catalog: Amazon Kinesis Data Streams and Amazon Kinesis Data Analytics for near-real-time data insights; Amazon Kinesis Data Firehose for clickstream analytics; AWS Glue to generate personalized product recommendations
- AWS Glue as the data catalog; Amazon Kinesis Data Streams and Amazon Kinesis Data Analytics for historical data insights; Amazon Kinesis Data Firehose for delivery to Amazon ES for clickstream analytics; Amazon EMR to generate personalized product recommendations
- Amazon Athena as the data catalog; Amazon Kinesis Data Streams and Amazon Kinesis Data Analytics for historical data insights; Amazon DynamoDB streams for clickstream analytics; AWS Glue to generate personalized product recommendations

Explanation:
The optimal solution leverages a combination of AWS services, each tailored for specific data ingestion and analytics needs. AWS Glue serves as the central data catalog, providing a metadata repository that allows various services to discover and understand the data stored in S3. Kinesis Data Streams is ideal for ingesting real-time data streams, and Kinesis Data Analytics provides the capability to process and analyze these streams in real-time, supporting immediate insights. Kinesis Data Firehose is well-suited for capturing and loading clickstream data into Amazon Elasticsearch Service (Amazon ES), enabling clickstream analytics. Amazon EMR offers a scalable platform for running big data processing frameworks like Spark and Hadoop, crucial for generating personalized product recommendations, a computationally intensive task.
Option B incorrectly uses Athena as the data catalog; while Athena allows querying data in S3, Glue is the preferred service for managing metadata and enabling data discovery across various AWS services. Furthermore, Glue is not typically used to directly generate personalized product recommendations. Option C incorrectly associates Kinesis Data Streams and Analytics with historical data insights. These services are designed for real-time processing, not historical analysis. Option D makes incorrect use of DynamoDB Streams for clickstream analytics. DynamoDB Streams are typically for tracking changes to data in DynamoDB tables, not for ingesting high-velocity clickstream data. Also, Athena as catalog is not ideal.
Therefore, option A offers the most appropriate combination of services to address all the requirements.
Supporting Links:

AWS Glue: https://aws.amazon.com/glue/
Amazon Kinesis Data Streams: https://aws.amazon.com/kinesis/data-streams/
Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/
Amazon EMR: https://aws.amazon.com/emr/

Question: 38

Exam Heist

A company is observing low accuracy while training on the default built-in image classification algorithm in Amazon SageMaker. The Data Science team wants to use an Inception neural network architecture instead of a ResNet architecture.
Which of the following will accomplish this? (Choose two.)

- [ ] Customize the built-in image classification algorithm to use Inception and use this for model training.
- [ ] Create a support case with the SageMaker team to change the default image classification algorithm to Inception.
- [ ] Bundle a Docker container with TensorFlow Estimator loaded with an Inception network and use this for model training. ✓
- [ ] Use custom code in Amazon SageMaker with TensorFlow Estimator to load the model with an Inception network, and use this for model training. ✓
- [ ] Download and apt-get install the inception network code into an Amazon EC2 instance and use this instance as a Jupyter notebook in Amazon SageMaker.

Explanation:
The question addresses adapting the Amazon SageMaker built-in image classification algorithm when it yields low accuracy and the Data Science team prefers an Inception neural network architecture over the default ResNet.
Option C is correct because bundling a Docker container offers a high degree of customization and control over the training environment. By including a TensorFlow Estimator loaded with an Inception network, the team can directly define and train the model using the desired architecture. This approach ensures compatibility and avoids limitations of the built-in algorithm. [Docker & SageMaker integration: https://docs.aws.amazon.com/sagemaker/latest/dg/docker-container.html]
Option D is also correct. Using custom code within Amazon SageMaker with TensorFlow Estimator gives a level of flexibility similar to using a Docker container but might be easier to implement initially. The team can define the Inception network architecture using TensorFlow within a SageMaker script, leverage SageMaker's infrastructure for training, and manage dependencies easily. This aligns well with the company's requirement to move away from the built-in algorithm and train the model using Inception. [SageMaker with TensorFlow: https://aws.amazon.com/blogs/machine-learning/bring-your-own-tensorflow-model-to-amazon-sagemaker/]
Option A is incorrect because customizing the built-in image classification algorithm beyond its intended parameters can be difficult or impossible. The built-in algorithms are designed with specific architectures in mind, and attempting to fundamentally alter them might not be supported.
Option B is incorrect. AWS support doesn't alter the default behaviors of pre-built algorithms based on user preferences. Their support focuses on assisting with usage and troubleshooting existing features.

Option E is incorrect because directly installing the Inception network code into an EC2 instance and using it as a Jupyter notebook within SageMaker doesn't directly integrate the Inception model with SageMaker's training and deployment functionalities. While it enables experimentation, it doesn't leverage SageMaker's scalability and managed environment for training.

Question: 39

Exam Heist

A Machine Learning Specialist built an image classification deep learning model. However, the Specialist ran into an overfitting problem in which the training and testing accuracies were 99% and 75%, respectively.
How should the Specialist address this issue and what is the reason behind it?

- The learning rate should be increased because the optimization process was trapped at a local minimum.
- The dropout rate at the flatten layer should be increased because the model is not generalized enough. ✓
- The dimensionality of dense layer next to the flatten layer should be increased because the model is not complex enough.
- The epoch number should be increased because the optimization process was terminated before it reached the global minimum.

Explanation:

The correct answer is B: Increase the dropout rate at the flatten layer.
Overfitting, indicated by high training accuracy (99%) and significantly lower testing accuracy (75%), signifies that the model is memorizing the training data rather than learning generalizable patterns. It's fitting the noise in the training data.
Why B is correct:

Dropout: Dropout is a regularization technique used in neural networks. It randomly disables a fraction of neurons during each training iteration. This forces the network to learn more robust and independent features, preventing over-reliance on specific neurons. It essentially creates multiple "thinned" versions of the network.
Flatten Layer: The flatten layer in a convolutional neural network (CNN) transforms the multi-dimensional feature maps from the convolutional layers into a one-dimensional vector before feeding them into the fully connected (dense) layers. This layer is a key transition point where complex features extracted by the convolutional layers are used for classification. A high dropout rate at the flatten layer helps prevent overfitting by forcing the subsequent dense layers to learn from a more diverse set of features.
Generalization: By increasing the dropout rate, the model is forced to become more generalized. It becomes less sensitive to specific features in the training data and more capable of performing well on unseen data. This is exactly what's needed to address the overfitting issue.
Why other options are incorrect:

A (Increasing Learning Rate): Increasing the learning rate can sometimes help escape local minima, but it more often exacerbates overfitting. A high learning rate can cause the model to bounce around and miss the optimal solution, making it less stable.
C (Increasing Dimensionality of Dense Layer): Increasing the dimensionality of a dense layer makes the model more complex. A more complex model is more likely to overfit, not less.
D (Increasing Epoch Number): Increasing the epoch number will train the model for longer. If the model is already overfitting, training for longer will only worsen the problem. It will overfit even more to the training data.
Supporting Cloud Concepts:
Deep learning models, particularly for image classification, require substantial computational resources. AWS provides various services like EC2 instances with GPUs or specialized machine learning services like SageMaker to train these models. SageMaker also supports various regularization techniques and hyperparameter tuning, which are crucial in addressing overfitting issues like this.
Authoritative Links:

Dropout Regularization: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*
https://jmlr.org/papers/v15/srivastava14a.html
Overfitting: https://www.ibm.com/docs/en/cloud-private/3.1.2?topic=terms-overfitting
AWS SageMaker: https://aws.amazon.com/sagemaker/

Question: 40

Exam Heist

A Machine Learning team uses Amazon SageMaker to train an Apache MXNet handwritten digit classifier model using a research dataset. The team wants to receive a notification when the model is overfitting. Auditors want to view the Amazon SageMaker log activity report to ensure there are no

unauthorized API calls.
What should the Machine Learning team do to address the requirements with the least amount of code and fewest steps?

- Implement an AWS Lambda function to log Amazon SageMaker API calls to Amazon S3. Add code to push a custom metric to Amazon CloudWatch. Create an alarm in CloudWatch with Amazon SNS to receive a notification when the model is overfitting.
- Use AWS CloudTrail to log Amazon SageMaker API calls to Amazon S3. Add code to push a custom metric to Amazon CloudWatch. Create an alarm in CloudWatch with Amazon SNS to receive a notification when the model is overfitting. ✓
- Implement an AWS Lambda function to log Amazon SageMaker API calls to AWS CloudTrail. Add code to push a custom metric to Amazon CloudWatch. Create an alarm in CloudWatch with Amazon SNS to receive a notification when the model is overfitting.
- Use AWS CloudTrail to log Amazon SageMaker API calls to Amazon S3. Set up Amazon SNS to receive a notification when the model is overfitting

Explanation:
The correct answer is B. Here's why:

Auditing API Calls (Unauthorized API Calls): AWS CloudTrail is the best service for logging API calls made to AWS services, including SageMaker. It provides an audit trail of who did what, when, and from where. Implementing a Lambda function (options A and C) for this purpose would be redundant and less efficient than using CloudTrail, which is specifically designed for this.
Overfitting Notification: The question requires notification upon detecting overfitting. The best way to detect overfitting is to monitor model performance metrics (e.g., validation loss) during training.
Custom Metric in CloudWatch: The proposed solution involves pushing custom metrics (specifically, metrics indicating overfitting) to Amazon CloudWatch. This is essential for tracking model performance.
CloudWatch Alarm & SNS: A CloudWatch alarm can be configured to trigger when a specified metric crosses a threshold, indicating overfitting. By connecting this alarm to an Amazon SNS topic, the team can receive notifications when the overfitting condition is met.
Let's analyze why the other options are less suitable:

Option A & C (Using Lambda to log API Calls): Both options A and C suggest using Lambda for logging SageMaker API calls. This introduces unnecessary complexity, code, and operational overhead when AWS CloudTrail is the native and recommended solution for this purpose. CloudTrail is a fully managed service specifically designed for logging API activity, making it the simplest and most efficient approach. Option C also proposes logging to CloudTrail using Lambda, which is logically incorrect as CloudTrail already captures the logs directly.
Option D (Missing Custom Metric Push): Option D lacks the critical step of *pushing a custom metric* that indicates overfitting to CloudWatch. Without this metric, a CloudWatch alarm cannot be created to trigger notifications related to overfitting. Simply logging API calls to S3 doesn't address the requirement of being notified about overfitting during training.
In conclusion, Option B effectively addresses both the auditing requirements and the overfitting notification requirement with the least amount of code and fewest steps. It leverages native AWS services like CloudTrail and CloudWatch, which are designed for these specific purposes.
Supporting Links:

AWS CloudTrail: https://aws.amazon.com/cloudtrail/
Amazon CloudWatch: https://aws.amazon.com/cloudwatch/
Amazon SNS: https://aws.amazon.com/sns/

Question: 41

Exam Heist

A Machine Learning Specialist is building a prediction model for a large number of features using linear models, such as linear regression and logistic regression.
During exploratory data analysis, the Specialist observes that many features are highly correlated with each other. This may make the model unstable.
What should be done to reduce the impact of having such a large number of features?

- Perform one-hot encoding on highly correlated features.
- Use matrix multiplication on highly correlated features.
- Create a new feature space using principal component analysis (PCA) ✓
- Apply the Pearson correlation coefficient.

Explanation:
The correct answer is C: Create a new feature space using principal component analysis (PCA). Here's why:

The problem describes multicollinearity, where features are highly correlated. Multicollinearity in linear models like linear and logistic regression leads to unstable coefficient estimates. This means the model's parameters become sensitive to small changes in the data, making it difficult to interpret feature importance and potentially harming the model's generalization performance.
PCA addresses this by transforming the original features into a new set of uncorrelated features called principal components. These components are linear combinations of the original features, ordered by the amount of variance they explain in the data. By selecting the top principal components, the specialist can reduce the dimensionality of the feature space while retaining most of the important information. This reduces multicollinearity, stabilizes the model, and can improve its performance.
Option A, one-hot encoding, expands categorical features into multiple binary features. While useful for handling categorical data, it would actually *increase* the number of features, exacerbating the multicollinearity problem, not solving it. It's not relevant to correlated *numerical* features, which is the core issue.
Option B, using matrix multiplication on highly correlated features, does not address multicollinearity. Matrix multiplication is a fundamental linear algebra operation but offers no inherent solution to feature correlation. It might even amplify the problem depending on how it's applied.
Option D, applying the Pearson correlation coefficient, is a *diagnostic* tool for detecting multicollinearity, not a solution. It helps identify which features are correlated, but it doesn't reduce the number of features or address the model instability caused by the correlation.
PCA effectively mitigates multicollinearity by creating uncorrelated features, reducing the dimensionality of the feature space, and stabilizing the model.
Further reading:

Principal Component Analysis (PCA): https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Multicollinearity: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/multicollinearity/

Question: 42

Exam Heist

A Machine Learning Specialist is implementing a full Bayesian network on a dataset that describes public transit in New York City. One of the random variables is discrete, and represents the number of minutes New Yorkers wait for a bus given that the buses cycle every 10 minutes, with a mean of 3 minutes.
Which prior probability distribution should the ML Specialist use for this variable?

- Poisson distribution ✓
- Uniform distribution
- Normal distribution
- Binomial distribution

Explanation:
The Poisson distribution is the most suitable prior probability distribution for the given scenario due to its properties relating to count data and waiting times in contexts like public transportation.
The random variable represents the number of minutes New Yorkers wait for a bus. This is essentially counting the occurrences of a specific event (the bus arriving) within a given interval (10 minutes cycle). The Poisson distribution models the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known average rate and independently of the time since the last event. The mean of 3 minutes aligns directly with the expected value (lambda) of the Poisson distribution.
While the uniform distribution could represent a random waiting time within the 10-minute cycle, it doesn't incorporate the provided information about the mean waiting time. The normal distribution, while common, is best suited for continuous variables and is not ideal for modeling discrete counts of waiting time. The binomial distribution models the number of successes in a fixed number of trials. While related to counts, it does not directly model waiting times like the Poisson distribution, which captures the frequency of events within an interval. The Poisson distribution, characterized by its single parameter (lambda) representing the average rate, can be easily updated with new data through Bayesian inference, making it an appropriate choice for a prior.
Therefore, the Poisson distribution is most appropriate for modeling the number of minutes New Yorkers wait for a bus, given a mean waiting time, as it models count data representing events within a specified interval, such as a bus cycle.https://en.wikipedia.org/wiki/Poisson_distributionhttps://www.statisticshowto.com/probability-and-statistics/distributions/poisson-distribution/

Question: 43

Exam Heist

A Data Science team within a large company uses Amazon SageMaker notebooks to access data stored in Amazon S3 buckets. The IT Security team is concerned that internet-enabled notebook instances create a security vulnerability where malicious code running on the instances could compromise data privacy.
The company mandates that all instances stay within a secured VPC with no internet access, and data communication traffic must stay within the AWS network.
How should the Data Science team configure the notebook instance placement to meet these requirements?

- Associate the Amazon SageMaker notebook with a private subnet in a VPC. Place the Amazon SageMaker endpoint and S3 buckets within the same VPC.
- Associate the Amazon SageMaker notebook with a private subnet in a VPC. Use IAM policies to grant access to Amazon S3 and Amazon SageMaker.
- Associate the Amazon SageMaker notebook with a private subnet in a VPC. Ensure the VPC has S3 VPC endpoints and Amazon SageMaker VPC endpoints attached to it. ✓
- Associate the Amazon SageMaker notebook with a private subnet in a VPC. Ensure the VPC has a NAT gateway and an associated security group allowing only outbound connections to Amazon S3 and Amazon SageMaker.

Explanation:
The correct answer is C. To meet the security requirements of a company mandating no internet access for its Amazon SageMaker notebook instances and ensuring all data communication stays within the AWS network, you need to place the notebook instance in a private subnet of a VPC and configure VPC endpoints for both Amazon S3 and SageMaker.
Option C fulfills all the requirements. Associating the notebook with a private subnet ensures it doesn't have a direct public IP address, thus isolating it from the internet. Using S3 VPC endpoints allows the notebook instance, through the AWS network, to access data stored in S3 buckets without traversing the public internet. Similarly, SageMaker VPC endpoints ensure that calls to SageMaker APIs (like training jobs or model deployments) also remain within the AWS network, eliminating any external internet dependencies.
Option A lacks the essential VPC endpoints. While placing the notebook, endpoint, and S3 within the same VPC is good practice, it doesn't guarantee the traffic will remain within the AWS network without VPC endpoints. The notebook instance would still try to use public internet to reach S3 without an S3 VPC endpoint.
Option B is incomplete. IAM policies grant access, but they don't force the traffic to stay within the AWS network. IAM policies authorize access, but VPC endpoints control network routing. While IAM policies are still needed for proper authorization, they are not sufficient by themselves.
Option D introduces a NAT gateway which defeats the purpose of blocking internet access. A NAT gateway allows instances in a private subnet to initiate outbound internet connections. This opens up the same security vulnerability the company is trying to avoid. Even with security group restrictions, a NAT gateway inherently permits external connections, violating the core requirement.
Therefore, using VPC endpoints for S3 and SageMaker combined with a private subnet creates a fully isolated environment where the notebook can securely access data and services while adhering to the company's security policy.
Relevant documentation:

VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
SageMaker Notebook Instances in a VPC: https://docs.aws.amazon.com/sagemaker/latest/dg/notebook-vpc.html

Question: 44

Exam Heist

A Machine Learning Specialist has created a deep learning neural network model that performs well on the training data but performs poorly on the test data.
Which of the following methods should the Specialist consider using to correct this? (Choose three.)

- [ ] Decrease regularization.
- [x] Increase regularization. ✓
- [x] Increase dropout. ✓
- [ ] Decrease dropout.
- [ ] Increase feature combinations.
- [x] Decrease feature combinations. ✓

Explanation:
The scenario describes overfitting, where the model memorizes the training data instead of learning generalizable patterns. The goal is to improve the model's performance on unseen data (the test set).

B. Increase regularization: Regularization techniques (L1, L2) add a penalty to the model's complexity, discouraging it from learning the noise in the training data and promoting simpler, more generalizable models. This combats overfitting. (Source: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l1-regularization)
C. Increase dropout: Dropout is a regularization technique specific to neural networks. During training, it randomly deactivates some neurons. This prevents neurons from becoming overly reliant on specific features or other neurons, forcing the network to learn more robust and independent representations, thereby reducing overfitting. (Source: https://jmlr.org/v15/srivastava14a.html)
F. Decrease feature combinations: Creating too many complex feature combinations can lead to the model fitting the training data very closely, including its noise. This results in overfitting. Reducing the number of feature combinations simplifies the model and makes it more likely to generalize well to unseen data. Overly complex feature engineering, while helpful in some cases, can be a source of overfitting if not carefully managed. By reducing feature combinations, you are essentially reducing the complexity of the model.
Why the other options are incorrect:

A. Decrease regularization: Decreasing regularization would likely exacerbate overfitting, allowing the model to become even more complex and memorize the training data better.
D. Decrease dropout: Decreasing dropout reduces its regularization effect, which can further promote overfitting in neural networks.
E. Increase feature combinations: Increasing feature combinations leads to a more complex model, potentially overfitting the training data.

Question: 45

Exam Heist

A Data Scientist needs to create a serverless ingestion and analytics solution for high-velocity, real-time streaming data.
The ingestion process must buffer and convert incoming records from JSON to a query-optimized, columnar format without data loss. The output datastore must be highly available, and Analysts must be able to run SQL queries against the data and connect to existing business intelligence dashboards.
Which solution should the Data Scientist build to satisfy the requirements?

- Create a schema in the AWS Glue Data Catalog of the incoming data format. Use an Amazon Kinesis Data Firehose delivery stream to stream the data and transform the data to Apache Parquet or ORC format using the AWS Glue Data Catalog before delivering to Amazon S3. Have the Analysts query the data directly from Amazon S3 using Amazon Athena, and connect to BI tools using the Athena Java Database Connectivity (JDBC) connector. ✓
- Write each JSON record to a staging location in Amazon S3. Use the S3 Put event to trigger an AWS Lambda function that transforms the data into Apache Parquet or ORC format and writes the data to a processed data location in Amazon S3. Have the Analysts query the data directly from Amazon S3 using Amazon Athena, and connect to BI tools using the Athena Java Database Connectivity (JDBC) connector.
- Write each JSON record to a staging location in Amazon S3. Use the S3 Put event to trigger an AWS Lambda function that transforms the data into Apache Parquet or ORC format and inserts it into an Amazon RDS PostgreSQL database. Have the Analysts query and run dashboards from the RDS database.
- Use Amazon Kinesis Data Analytics to ingest the streaming data and perform real-time SQL queries to convert the records to Apache Parquet before delivering to Amazon S3. Have the Analysts query the data directly from Amazon S3 using Amazon Athena and connect to BI tools using the Athena Java Database Connectivity (JDBC) connector.

Explanation:
The correct answer is A. Here's why:
A is the most suitable solution because:

Kinesis Data Firehose for Real-time Ingestion: Kinesis Data Firehose is designed specifically for real-time streaming data ingestion into AWS data stores. It handles high velocity data effectively.
Buffering: Firehose buffers incoming records, fulfilling the requirement of buffering.
JSON to Columnar Format Conversion: Firehose can transform JSON data to a query-optimized columnar format like Parquet or ORC using AWS Glue Data Catalog for schema discovery and transformation.
Serverless: Kinesis Data Firehose is a managed service, which means serverless.
AWS Glue Data Catalog: Glue Data Catalog centrally stores metadata (schema definitions) making data discovery and querying easier.
Amazon S3 for Storage: S3 provides highly available and durable storage for the transformed data.
Amazon Athena for SQL Querying: Athena allows analysts to run SQL queries directly against the data in S3 without the need for a database. This fulfills the SQL query requirement.
BI Tool Connectivity: Athena's JDBC connector enables seamless integration with existing business intelligence (BI) dashboards.
Why the other options are less suitable:

B: Lambda-based Transformation: While Lambda can transform data, using Firehose is more efficient and scalable for high-velocity streaming data. S3 Put event triggering Lambda adds unnecessary complexity. Lambda can be used for complex transformations that Kinesis Data Firehose cannot handle.
C: RDS PostgreSQL: Using RDS PostgreSQL as the final datastore is not ideal for large-scale analytical workloads. Columnar formats in S3 queried by Athena are better suited for this. Also, inserting individual records into RDS is not as efficient.
D: Kinesis Data Analytics for Format Conversion: While Kinesis Data Analytics can perform real-time SQL, it is primarily for real-time data processing and analytics, not optimal for format conversion at this scale and for storage in a columnar format. It is less cost-effective for simple format conversion.
Supporting Links:

Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/
AWS Glue Data Catalog: https://aws.amazon.com/glue/
Amazon Athena: https://aws.amazon.com/athena/

Question: 46

Exam Heist

An online reseller has a large, multi-column dataset with one column missing 30% of its data. A Machine Learning Specialist believes that certain columns in the dataset could be used to reconstruct the missing data.
Which reconstruction approach should the Specialist use to preserve the integrity of the dataset?

- Listwise deletion
- Last observation carried forward
- Multiple imputation ✓
- Mean substitution

Explanation:
The question concerns handling missing data in a machine learning context to preserve dataset integrity, specifically when some features might predict the missing values. Option C, multiple imputation, is the most suitable approach in this scenario. Here's why: Multiple imputation involves creating multiple plausible datasets, each with different imputed values for the missing data. The imputation is based on the relationships between the missing variable and other observed variables in the dataset. This effectively leverages the "certain columns" the Specialist suspects can reconstruct missing data. By generating multiple imputed datasets, we acknowledge the uncertainty surrounding the missing values and avoid introducing bias by selecting a single "best guess." This approach leverages the predictive power of other columns to create values for the missing ones.
Option A, listwise deletion, removes entire rows containing missing values. This leads to significant data loss (30% in this case) and can introduce bias if the missingness is not completely random. It ignores the potential to reconstruct the missing data using other features.
Option B, last observation carried forward (LOCF), is primarily used in time-series data where the last known value is assumed to be a reasonable estimate for the missing value. It's inappropriate for general datasets without a temporal component and wouldn't utilize relationships with other columns for reconstruction.
Option D, mean substitution, replaces missing values with the mean of the available data for that column. While simple, it reduces variance and can distort the distribution of the variable, leading to biased statistical analyses and inaccurate machine learning models. It doesn't leverage information from other columns to predict the missing values.
Multiple imputation, on the other hand, maintains the relationships between variables and provides a more accurate representation of the data's distribution, preventing bias. It considers all observed data, therefore adhering to the principle of using other available columns to impute and fill in data. By generating several imputed datasets, it enables better estimation of uncertainty.
Therefore, multiple imputation is the superior choice for reconstructing missing data while preserving the integrity and statistical properties of the dataset in a machine learning context. It's crucial when relationships between variables hold vital information for accurate data imputation.
Relevant Links:

Multiple Imputation: https://en.wikipedia.org/wiki/Multiple_imputation
Handling Missing Data: https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1c6929c79431

Question: 47

Exam Heist

A company is setting up an Amazon SageMaker environment. The corporate data security policy does not allow communication over the internet. How can the company enable the Amazon SageMaker service without enabling direct internet access to Amazon SageMaker notebook instances?

- Create a NAT gateway within the corporate VPC.
- Route Amazon SageMaker traffic through an on-premises network.
- Create Amazon SageMaker VPC interface endpoints within the corporate VPC. ✓
- Create VPC peering with Amazon VPC hosting Amazon SageMaker.

Explanation:

The correct answer is C: Create Amazon SageMaker VPC interface endpoints within the corporate VPC. This approach allows SageMaker notebook instances to communicate with the SageMaker service through AWS's internal network, completely avoiding the public internet.

VPC interface endpoints, powered by AWS PrivateLink, provide private connectivity to AWS services and supported VPC endpoint services from within your VPC, without exposing your traffic to the public internet. By creating SageMaker VPC interface endpoints (specifically for SageMaker API, SageMaker Runtime, and SageMaker Notebook Instances), you establish a secure, direct connection between your notebook instances and the SageMaker service, all within the boundaries of your VPC.

Option A, creating a NAT Gateway, enables outbound internet access for instances in a private subnet. This would violate the corporate data security policy prohibiting internet communication.

Option B, routing traffic through an on-premises network, would require exposing SageMaker traffic to the internet or establishing a complex VPN/Direct Connect setup which also requires internet access as an intermediate hop, complicating security and adding latency and overhead.

Option D, VPC Peering, connects two VPCs, but it doesn't inherently solve the problem of internet access. While VPC Peering allows instances in the VPCs to communicate, it doesn't automatically create a private pathway to the SageMaker service. It still depends on routing and could potentially require internet access.

Therefore, using VPC interface endpoints is the most secure and efficient method to enable SageMaker access without internet access, aligning with the company's strict data security policy.

Here are some authoritative links for further research:

AWS PrivateLink: https://aws.amazon.com/privatelink/
Using Amazon SageMaker with VPC: https://docs.aws.amazon.com/sagemaker/latest/dg/vpc.html
Create VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/privatelink/create-endpoint.html

Question: 48

Exam Heist

A Machine Learning Specialist is training a model to identify the make and model of vehicles in images. The Specialist wants to use transfer learning and an existing model trained on images of general objects. The Specialist collated a large custom dataset of pictures containing different vehicle makes and models.

What should the Specialist do to initialize the model to re-train it with the custom data?

- Initialize the model with random weights in all layers including the last fully connected layer.
- Initialize the model with pre-trained weights in all layers and replace the last fully connected layer. ✓
- Initialize the model with random weights in all layers and replace the last fully connected layer.
- Initialize the model with pre-trained weights in all layers including the last fully connected layer.

Explanation:

The correct approach to transfer learning in this scenario, where a pre-trained model on general object images is adapted for identifying vehicle makes and models, is option B: "Initialize the model with pre-trained weights in all layers and replace the last fully connected layer."

Here's why:

Leveraging Pre-trained Knowledge: Transfer learning exploits knowledge gained from training on a large dataset (in this case, general object images) to improve performance on a related, but different task (vehicle identification). The early layers of a convolutional neural network (CNN) learn general features like edges, textures, and shapes. These features are useful regardless of whether the image contains a cat, a dog, or a car. Therefore, initializing with pre-trained weights allows the model to start with a strong foundation, reducing the need to learn these basic features from scratch on the new dataset.

The Role of the Last Fully Connected Layer: The last fully connected layer of the pre-trained model is specifically designed to classify images into the categories it was originally trained on (e.g., cats, dogs, birds, etc.). These categories are irrelevant to the vehicle identification task. Therefore, this layer needs to be replaced with a new fully connected layer whose output size matches the number of vehicle makes and models in the custom dataset.

Why not Random Initialization for all Layers? Initializing all layers with random weights (options A and C) defeats the purpose of transfer learning. It effectively means training a new model from scratch, which is computationally expensive and requires a large amount of labeled data to achieve good performance. The whole point of transfer learning is to avoid this.
Why not using pre-trained weights in all layers including the last one (option D)? This will cause a classification based on the initial set of labels, which is completely unrelated to the target vehicle identification task. You can't directly use the pre-trained last layer for new classes as the features space of pre-trained classes is different from the one required for the target task.

In summary: By initializing with pre-trained weights in most layers and replacing only the last fully connected layer, the model can quickly adapt to the vehicle identification task by fine-tuning the pre-trained features and learning new, task-specific features in the final layer.
Authoritative Links:

Transfer Learning: https://papers.nips.cc/paper/2009/file/f93cd75372f03442ab92f679fbb2e6cd-Paper.pdf
How transferable are features in deep neural networks? https://arxiv.org/abs/1411.1792

Question: 49

Exam Heist

An office security agency conducted a successful pilot using 100 cameras installed at key locations within the main office. Images from the cameras were uploaded to Amazon S3 and tagged using Amazon Rekognition, and the results were stored in Amazon ES. The agency is now looking to expand the pilot into a full production system using thousands of video cameras in its office locations globally. The goal is to identify activities performed by non-employees in real time
Which solution should the agency consider?

- Use a proxy server at each local office and for each camera, and stream the RTSP feed to a unique Amazon Kinesis Video Streams video stream. On each stream, use Amazon Rekognition Video and create a stream processor to detect faces from a collection of known employees, and alert when non-employees are detected. ✓
- Use a proxy server at each local office and for each camera, and stream the RTSP feed to a unique Amazon Kinesis Video Streams video stream. On each stream, use Amazon Rekognition Image to detect faces from a collection of known employees and alert when non-employees are detected.
- Install AWS DeepLens cameras and use the DeepLens_Kinesis_Video module to stream video to Amazon Kinesis Video Streams for each camera. On each stream, use Amazon Rekognition Video and create a stream processor to detect faces from a collection on each stream, and alert when non-employees are detected.
- Install AWS DeepLens cameras and use the DeepLens_Kinesis_Video module to stream video to Amazon Kinesis Video Streams for each camera. On each stream, run an AWS Lambda function to capture image fragments and then call Amazon Rekognition Image to detect faces from a collection of known employees, and alert when non-employees are detected.

Explanation:
Here's a detailed justification for why option A is the most suitable solution for the office security agency's requirements, along with supporting concepts and links:
The primary goal is real-time identification of activities performed by non-employees using thousands of video cameras globally. This demands a solution capable of handling high-volume, continuous video streams and performing real-time analysis.

Kinesis Video Streams: This service is specifically designed to ingest, store, and process video streams at scale. It efficiently handles the continuous stream of video data from numerous cameras. https://aws.amazon.com/kinesis/video-streams/
RTSP and Proxy Servers: Real-Time Streaming Protocol (RTSP) is a common protocol for streaming video from IP cameras. Using a proxy server at each local office is beneficial for several reasons: it can reduce bandwidth consumption by caching frequently accessed content, improve security by hiding the internal network structure, and distribute the load of streaming video. Each camera streaming to a unique Kinesis Video Stream gives granular control and parallelism.
Amazon Rekognition Video: This service provides real-time facial analysis capabilities within video streams. It allows you to detect faces, compare them against a collection of known faces (employees), and identify non-employees. It provides real-time analysis. https://aws.amazon.com/rekognition/video/
Stream Processor: A Rekognition Video stream processor enables continuous analysis of the video stream for specific events (in this case, non-employee detection). It automates the process of facial recognition and alerting.
Why other options are less suitable:

Option B (Amazon Rekognition Image): Rekognition Image analyzes still images, not video streams in real time. Analyzing individual frames extracted from a video stream would be less efficient and miss crucial temporal information (like movement patterns).
Option C and D (AWS DeepLens): While DeepLens is useful for edge computing and machine learning at the device level, it might not be the most cost-effective solution for thousands of cameras. It introduces complexity by adding device management. The question emphasizes expansion of an existing system that uploads to S3; DeepLens is a different approach

that might not integrate easily.
Option D (AWS Lambda): Using a Lambda function to capture image fragments and then call Rekognition Image would introduce latency and complexity, making real-time analysis challenging to achieve. Lambda is also not suited for continuous stream processing in this manner; Rekognition Video is a dedicated service for this specific purpose.
Therefore, Option A provides the most efficient, scalable, and real-time solution for identifying non-employees in a global environment with thousands of video cameras by utilizing Kinesis Video Streams for ingestion and Rekognition Video for the facial recognition.

Question: 50

Exam Heist

A Marketing Manager at a pet insurance company plans to launch a targeted marketing campaign on social media to acquire new customers.
Currently, the company has the following data in Amazon Aurora:
- Profiles for all past and existing customers
- Profiles for all past and existing insured pets
- Policy-level information
- Premiums received
- Claims paid

What steps should be taken to implement a machine learning model to identify potential new customers on social media?

- ○ Use regression on customer profile data to understand key characteristics of consumer segments. Find similar profiles on social media
- ○ Use clustering on customer profile data to understand key characteristics of consumer segments. Find similar profiles on social media ✓
- ○ Use a recommendation engine on customer profile data to understand key characteristics of consumer segments. Find similar profiles on social media.
- ○ Use a decision tree classifier engine on customer profile data to understand key characteristics of consumer segments. Find similar profiles on social media.

Explanation:
The most appropriate approach to identify potential new customers on social media given the provided data and business objective is option B: "Use clustering on customer profile data to understand key characteristics of consumer segments. Find similar profiles on social media."
Here's why:

Understanding Consumer Segments: The core goal is to identify groups of customers with similar characteristics. Clustering algorithms are specifically designed for this purpose. They can automatically group customers based on shared attributes derived from their profiles (demographics, pet information, policy details, etc.). This allows the marketing manager to define distinct consumer segments. K-Means clustering is a popular algorithm for this.
Targeted Marketing: By understanding the common traits of profitable customer segments, the marketing manager can then search for similar profiles on social media platforms. This ensures the marketing campaign is targeted at individuals who are most likely to be interested in pet insurance.
Alternatives:

Regression (A): Regression is primarily used for predicting continuous values (e.g., customer lifetime value). While it could be used to predict customer behavior, it doesn't directly help in identifying distinct customer segments.
Recommendation Engine (C): Recommendation engines are designed to suggest items to users based on their past behavior. This is not the best fit for finding new customers on social media.
Decision Tree Classifier (D): Decision trees are used for classification problems, where you want to predict a categorical outcome (e.g., whether someone will buy a policy or not). While potentially useful after customer segments are identified, it's not the initial step in identifying segments.
Clustering is thus superior as it provides a natural way to group existing customers into segments, enabling targeted marketing efforts on social media to find new customers with similar profiles. This approach allows the pet insurance company to focus its resources on individuals most likely to convert.
Further Research:

Clustering: https://scikit-learn.org/stable/modules/clustering.html
K-Means Clustering: https://aws.amazon.com/blogs/machine-learning/k-means-clustering-with-amazon-sagemaker/
Targeted Marketing: https://www.oracle.com/uk/cx/marketing/what-is-targeted-marketing/

Question: 51

Exam Heist

A manufacturing company has a large set of labeled historical sales data. The manufacturer would like to predict how many units of a particular part should be produced each quarter.
Which machine learning approach should be used to solve this problem?

- Logistic regression
- Random Cut Forest (RCF)
- Principal component analysis (PCA)
- Linear regression ✓

Explanation:
Here's a detailed justification for choosing linear regression as the appropriate machine learning approach:
The problem describes a scenario where a manufacturing company wants to predict the *quantity* of units to produce. Since we are predicting a continuous numerical value (number of units), this is fundamentally a regression problem.
Let's evaluate each option:

A. Logistic Regression: Logistic regression is used for classification problems, where the goal is to predict a categorical outcome (e.g., spam or not spam, success or failure). It predicts the probability of a binary outcome. This is not relevant to predicting a continuous number of units.
B. Random Cut Forest (RCF): RCF is an unsupervised anomaly detection algorithm. It's used to identify data points that are significantly different from the rest of the dataset. This isn't about predicting future sales quantity. Its use case is specifically detecting anamolies, such as fraud detection.
C. Principal Component Analysis (PCA): PCA is a dimensionality reduction technique. It's used to reduce the number of variables in a dataset while preserving its essential information. While PCA can be used as a preprocessing step, it doesn't directly predict sales quantity. It is more related to data optimization.
D. Linear Regression: Linear regression models the relationship between a dependent variable (sales quantity) and one or more independent variables (historical sales data, time period). It aims to find the best-fitting linear equation to predict the dependent variable based on the independent variables. Given the historical data, it can forecast future sales quantity. It fits the definition of a regression problem which is predicting future sales quantity.
Therefore, **Linear Regression** is the most appropriate machine learning approach. It directly addresses the problem of predicting a continuous numerical value (the number of units to produce) based on labeled historical data.**Authoritative Links for further research:**

AWS Machine Learning Documentation: https://aws.amazon.com/machine-learning/
Linear Regression on AWS: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html
Random Cut Forest on AWS: https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html
PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Logistic Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Question: 52

Exam Heist

A financial services company is building a robust serverless data lake on Amazon S3. The data lake should be flexible and meet the following requirements:
- Support querying old and new data on Amazon S3 through Amazon Athena and Amazon Redshift Spectrum.
- Support event-driven ETL pipelines
- Provide a quick and easy way to understand metadata

Which approach meets these requirements?

- Use an AWS Glue crawler to crawl S3 data, an AWS Lambda function to trigger an AWS Glue ETL job, and an AWS Glue Data catalog to search and discover metadata. ✓
- Use an AWS Glue crawler to crawl S3 data, an AWS Lambda function to trigger an AWS Batch job, and an external Apache Hive metastore to search and discover metadata.
- Use an AWS Glue crawler to crawl S3 data, an Amazon CloudWatch alarm to trigger an AWS Batch job, and an AWS Glue Data Catalog to search and discover metadata.
- Use an AWS Glue crawler to crawl S3 data, an Amazon CloudWatch alarm to trigger an AWS Glue ETL job, and an external Apache Hive metastore to search and discover metadata.

Explanation:
The correct answer is A. Here's a detailed justification:

AWS Glue Crawler: An AWS Glue crawler is crucial for automatically discovering the schema of the data residing in the S3 data lake. It scans the data and infers the structure, creating metadata tables in the AWS Glue Data Catalog. This addresses the requirement of supporting querying data through Athena and Redshift Spectrum, which rely on metadata.
AWS Lambda Function: AWS Lambda is perfectly suited for event-driven ETL pipelines. It can be triggered by S3 events (e.g., new data arriving) and initiate an AWS Glue ETL job. This satisfies the requirement of an event-driven pipeline.
AWS Glue ETL Job: AWS Glue ETL jobs provide the core functionality for transforming and loading data. This ensures that the data can be processed and prepared for analysis and querying.
AWS Glue Data Catalog: The AWS Glue Data Catalog is a managed metadata repository. It acts as a central place to store and discover metadata about the data in the S3 data lake. This directly addresses the requirement for a quick and easy way to understand metadata. Athena and Redshift Spectrum both integrate with the Glue Data Catalog.
Why other options are not as suitable:

B: While AWS Glue crawler and Lambda function can be used to crawl S3 data and trigger Batch job. Using an external Apache Hive metastore adds unnecessary complexity and management overhead compared to using the AWS Glue Data Catalog, which is fully managed and integrated with other AWS services.
C & D: Using CloudWatch alarms to trigger ETL jobs is not event-driven in the truest sense. It relies on a schedule or metric threshold rather than directly reacting to data arriving in S3. AWS Lambda offers a more flexible and responsive solution.
Authoritative Links:

AWS Glue: https://aws.amazon.com/glue/
AWS Lambda: https://aws.amazon.com/lambda/
Amazon Athena: https://aws.amazon.com/athena/
Amazon Redshift Spectrum: https://aws.amazon.com/redshift/spectrum/

Question: 53

Exam Heist

A company's Machine Learning Specialist needs to improve the training speed of a time-series forecasting model using TensorFlow. The training is currently implemented on a single-GPU machine and takes approximately 23 hours to complete. The training needs to be run daily.
The model accuracy is acceptable, but the company anticipates a continuous increase in the size of the training data and a need to update the model on an hourly, rather than a daily, basis. The company also wants to minimize coding effort and infrastructure changes.
What should the Machine Learning Specialist do to the training solution to allow it to scale for future demand?

- Do not change the TensorFlow code. Change the machine to one with a more powerful GPU to speed up the training.
- Change the TensorFlow code to implement a Horovod distributed framework supported by Amazon SageMaker. Parallelize the training to as many machines as needed to achieve the business goals. ✓
- Switch to using a built-in AWS SageMaker DeepAR model. Parallelize the training to as many machines as needed to achieve the business goals.
- Move the training to Amazon EMR and distribute the workload to as many machines as needed to achieve the business goals.

Explanation:
The correct answer is B. Here's a detailed justification:
The problem requires reducing the training time of a TensorFlow time-series model, anticipating increasing data size, and a shift to hourly training updates while minimizing coding effort and infrastructure changes.
Option B, utilizing Horovod within Amazon SageMaker, directly addresses these needs. Horovod is a distributed deep learning training framework that simplifies the parallelization of TensorFlow models. SageMaker provides managed infrastructure and tooling that significantly reduces the effort required to set up and manage distributed training environments.
https://horovod.readthedocs.io/en/stable/ & https://aws.amazon.com/sagemaker/
Horovod's ease of integration with TensorFlow minimizes code changes, satisfying the "minimize coding effort" requirement. Scaling the training is achieved by parallelizing the workload across multiple machines provisioned by SageMaker, addressing the need for future scalability and faster training.
Option A, simply upgrading the GPU, might provide some initial improvement, but it's a short-term solution. It doesn't scale well as data increases and eventually hits hardware limitations. It also does not address the need to update the model hourly.
Option C, switching to SageMaker DeepAR, could be considered. However, it necessitates a complete model rewrite which increases the coding effort substantially, violating the "minimize coding effort" constraint. DeepAR is a viable solution when starting a time-series forecasting project from scratch, but not optimal when refactoring existing models.

Option D, moving the training to Amazon EMR, introduces a larger infrastructure change. EMR typically involves more complex cluster management and requires significant configuration compared to SageMaker's managed distributed training capabilities. This increases the operational overhead and defeats the need to minimize infrastructural changes. Additionally, while EMR supports distributed training, it doesn't offer the same seamless integration with TensorFlow and Horovod as SageMaker.

Question: 54

Exam Heist

Which of the following metrics should a Machine Learning Specialist generally use to compare/evaluate machine learning classification models against each other?

- Recall
- Misclassification rate
- Mean absolute percentage error (MAPE)
- Area Under the ROC Curve (AUC) ✓

Explanation:

The correct answer is **D. Area Under the ROC Curve (AUC)**. Here's why:

AUC is a comprehensive metric for evaluating classification models because it considers the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) across various classification thresholds. A ROC (Receiver Operating Characteristic) curve plots these rates, and AUC quantifies the area under this curve. A higher AUC (closer to 1) indicates a better-performing model, suggesting it effectively distinguishes between positive and negative classes. This makes AUC a valuable metric for model comparison, as it provides an aggregated measure of performance independent of a single chosen threshold.

A. **Recall** (also known as sensitivity or true positive rate) focuses only on the ability of the model to identify positive instances. While important, it doesn't consider false positives. A model can achieve perfect recall by simply predicting everything as positive, which is often undesirable.

B. **Misclassification rate** (or error rate) represents the proportion of incorrect predictions. While simple to understand, it can be misleading, especially with imbalanced datasets where one class significantly outnumbers the other. A model might have a low misclassification rate simply by predicting the majority class most of the time.

C. **Mean Absolute Percentage Error (MAPE)** is a common metric for regression problems, not classification. It measures the average percentage difference between predicted and actual continuous values, which is not relevant when evaluating the performance of classification models that predict categorical outcomes.

Therefore, AUC offers a more balanced and informative measure of classification model performance than recall or misclassification rate, and it is appropriate for classification tasks unlike MAPE. It considers both sensitivity and specificity, making it suitable for comparing models across different classification tasks, particularly where the class distribution might be uneven. Selecting a good threshold can be chosen based on a cost/benefit analysis.

Further reading:

Understanding AUC: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

ROC and AUC: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

Question: 55

Exam Heist

A company is running a machine learning prediction service that generates 100 TB of predictions every day. A Machine Learning Specialist must generate a visualization of the daily precision-recall curve from the predictions, and forward a read-only version to the Business team.
Which solution requires the LEAST coding effort?

- Run a daily Amazon EMR workflow to generate precision-recall data, and save the results in Amazon S3. Give the Business team read-only access to S3.
- Generate daily precision-recall data in Amazon QuickSight, and publish the results in a dashboard shared with the Business team.
- Run a daily Amazon EMR workflow to generate precision-recall data, and save the results in Amazon S3. Visualize the arrays in Amazon QuickSight, and publish them in a dashboard shared with the Business team. ✓
- Generate daily precision-recall data in Amazon ES, and publish the results in a dashboard shared with the Business team.

Explanation:

The question asks for the solution that requires the *least* coding effort to visualize daily precision-recall curves from 100 TB of daily predictions and share a read-only version with the business team.
Option C is the best answer because it leverages the strengths of both Amazon EMR and QuickSight with minimal coding. EMR is suitable for processing large datasets like 100TB to generate the precision-recall data. Then, QuickSight can visualize this data from S3 and publish an interactive dashboard for the business team with read-only access. QuickSight's ability to directly visualize data stored in S3 simplifies the process. This avoids complex data ingestion pipelines directly into visualization tools.
Option A requires only EMR and S3. It generates the precision-recall data and saves it to S3, granting the Business team read-only access. However, it doesn't natively provide a visualization, leaving the Business team to handle visualization themselves (which demands more coding or using other tools by the business team).
Option B suggests generating precision-recall data in QuickSight directly. While QuickSight excels at visualization, generating insights directly within QuickSight for 100TB of daily data would be computationally expensive and challenging. QuickSight is not designed to handle that data volume for real-time calculations without pre-aggregation. This would likely involve writing custom code or utilizing more complex QuickSight functionalities.
Option D suggests using Amazon ES. ES is a good choice for indexing and searching logs and operational data. Generating and visualizing 100TB of data in ES is not its core strength. ES would need to ingest the data, potentially index it, and then create visualizations. This would likely require more coding and is not optimized for numerical computation required for precision-recall curve generation.
Therefore, Option C provides the best balance between data processing and visualization with the *least* coding effort by utilizing EMR for data processing and QuickSight for visualization.
Supporting Links:

Amazon EMR: https://aws.amazon.com/emr/
Amazon QuickSight: https://aws.amazon.com/quicksight/
Amazon S3: https://aws.amazon.com/s3/

Question: 56

Exam Heist

A Machine Learning Specialist is preparing data for training on Amazon SageMaker. The Specialist is using one of the SageMaker built-in algorithms for the training. The dataset is stored in .CSV format and is transformed into a numpy.array, which appears to be negatively affecting the speed of the training.
What should the Specialist do to optimize the data for training on SageMaker?

- Use the SageMaker batch transform feature to transform the training data into a DataFrame.
- Use AWS Glue to compress the data into the Apache Parquet format.
- Transform the dataset into the RecordIO protobuf format. ✓
- Use the SageMaker hyperparameter optimization feature to automatically optimize the data.

Explanation:
The correct answer is **C. Transform the dataset into the RecordIO protobuf format.** Here's why:
The problem states that converting the CSV data to a NumPy array is slowing down the training process. SageMaker built-in algorithms are optimized to work with specific data formats, often more efficient than generic NumPy arrays for large datasets.

RecordIO/protobuf is a highly efficient binary format that SageMaker algorithms are designed to read quickly. This format reduces parsing overhead compared to CSV and optimizes data transfer to the training instances. Using RecordIO also allows efficient sharding of data for distributed training.
Let's analyze why the other options are less optimal:

A. Use the SageMaker batch transform feature to transform the training data into a DataFrame. Batch transform is primarily for inference (generating predictions on a large dataset), not optimizing the training data format. While DataFrames can be efficient, they are not the most efficient format for SageMaker built-in algorithms compared to RecordIO.
B. Use AWS Glue to compress the data into the Apache Parquet format. Parquet is a columnar storage format that's excellent for analytics and querying in services like Athena. While compression is beneficial, Parquet is not the most natively supported and optimized format for SageMaker *training* specifically. Furthermore, Glue is generally used for data transformation at rest, not specifically to accelerate the data pipeline to the training job.
D. Use the SageMaker hyperparameter optimization feature to automatically optimize the data. Hyperparameter optimization optimizes the *model's* parameters, not the data format. While hyperparameter optimization is a crucial part of machine learning, it doesn't address the data format inefficiency highlighted in the scenario.
In summary, RecordIO/protobuf format is the most suitable option because it is designed to improve the speed of the training process when using built-in SageMaker algorithms due to its efficient parsing and data transfer characteristics.

Here are authoritative links for further research:

SageMaker Data Formats: Outlines supported data formats for SageMaker.
SageMaker Built-in Algorithms: Provides details about individual algorithms and their expected data format.
Amazon SageMaker Examples: Contains many examples utilizing the RecordIO format for training.

Question: 57

Exam Heist

A Machine Learning Specialist is required to build a supervised image-recognition model to identify a cat. The ML Specialist performs some tests and records the following results for a neural network-based image classifier:
Total number of images available = 1,000
Test set images = 100 (constant test set)
The ML Specialist notices that, in over 75% of the misclassified images, the cats were held upside down by their owners.
Which techniques can be used by the ML Specialist to improve this specific test error?

- Increase the training data by adding variation in rotation for training images. ✓
- Increase the number of epochs for model training
- Increase the number of layers for the neural network.
- Increase the dropout rate for the second-to-last layer.

Explanation:
The correct answer is A: Increase the training data by adding variation in rotation for training images. Here's why:
The problem lies in the model's inability to generalize to images of cats in an unusual orientation (upside down). This indicates a lack of representation of rotated images in the training data, leading to poor performance on such examples in the test set.
Option A directly addresses this issue. By augmenting the training dataset with rotated images of cats, the model is exposed to the variations it's currently failing to recognize. This increased diversity in the training data enables the model to learn features that are invariant to rotation, improving its ability to correctly classify cats regardless of their orientation. Data augmentation is a common technique in machine learning to improve model generalization and robustness.
Option B, increasing the number of epochs, might help to a limited extent, but if the data lacks the necessary information (rotated images), the model will primarily overfit the existing data, leading to minimal improvements in the specific error case.
Option C, increasing the number of layers, increases the model's complexity and risk of overfitting if the training data isn't representative. It doesn't specifically address the lack of rotated images.
Option D, increasing the dropout rate, is a regularization technique that aims to prevent overfitting. While it might improve generalization slightly, it doesn't directly tackle the issue of the model's unfamiliarity with rotated images.
In conclusion, the most effective solution is to augment the training dataset with rotated cat images. This directly addresses the identified problem and enhances the model's ability to handle variations in cat orientation, thereby improving performance on the test set.
Further research:

Data Augmentation: https://www.tensorflow.org/tutorials/images/data_augmentation
Overfitting and Underfitting: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

Question: 58

Exam Heist

A Machine Learning Specialist needs to be able to ingest streaming data and store it in Apache Parquet files for exploration and analysis.
Which of the following services would both ingest and store this data in the correct format?

- AWS DMS
- Amazon Kinesis Data Streams
- Amazon Kinesis Data Firehose ✓
- Amazon Kinesis Data Analytics

Explanation:
The correct answer is **C. Amazon Kinesis Data Firehose**. Here's a detailed justification:
Kinesis Data Firehose is designed specifically for loading streaming data into data lakes and data warehouses. It can automatically convert streaming data into formats like Parquet before storing it in destinations such as Amazon S3. This fulfills the requirement of storing the ingested data in Parquet format. It can also handle data transformation, compression, and encryption while streaming.

Option A, AWS DMS (Database Migration Service), is used for migrating databases, not ingesting streaming data. It is geared toward transferring data from one database to another and does not directly handle streaming ingestion in Parquet.
Option B, Amazon Kinesis Data Streams, is for collecting and processing large streams of data records in real-time. While it can ingest data, it doesn't natively support converting and storing the data directly in Parquet format. You'd need additional processing, possibly using Kinesis Data Analytics, to transform and store the data in Parquet.
Option D, Amazon Kinesis Data Analytics, is for processing and analyzing streaming data in real time using SQL or Apache Flink. It doesn't directly ingest data or store it; instead, it performs computations on the data ingested by Kinesis Data Streams or Data Firehose. While it can be part of a solution, it's not the single service that both ingests and stores data in Parquet format.
Therefore, Kinesis Data Firehose stands out as the service that meets both ingestion and storage criteria, directly outputting Parquet format to destinations such as S3.
Authoritative Links:

Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/
Parquet Format: https://parquet.apache.org/

Question: 59

Exam Heist

A data scientist has explored and sanitized a dataset in preparation for the modeling phase of a supervised learning task. The statistical dispersion can vary widely between features, sometimes by several orders of magnitude. Before moving on to the modeling phase, the data scientist wants to ensure that the prediction performance on the production data is as accurate as possible.
Which sequence of steps should the data scientist take to meet these requirements?

- Apply random sampling to the dataset. Then split the dataset into training, validation, and test sets.
- Split the dataset into training, validation, and test sets. Then rescale the training set and apply the same scaling to the validation and test sets. ✓
- Rescale the dataset. Then split the dataset into training, validation, and test sets.
- Split the dataset into training, validation, and test sets. Then rescale the training set, the validation set, and the test set independently.

Explanation:
Here's a detailed justification for why option B is the correct approach, along with explanations of why the other options are less suitable:
The goal is to handle features with varying statistical dispersion and ensure good prediction performance in production. This requires splitting the data into training, validation, and test sets *before* rescaling to prevent data leakage. Data leakage occurs when information from the validation or test sets inadvertently influences the training process, leading to overly optimistic performance estimates during development that do not generalize well to unseen production data.
Option B: Correct Approach
Split the data: Divide the dataset into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune hyperparameters and evaluate model performance during training, and the test set is used for a final, unbiased evaluation of the model's generalization ability.
Rescale the training set: Apply a scaling technique (e.g., standardization or normalization) to the training set. Standardization transforms the data to have zero mean and unit variance, while normalization scales the data to a range between 0 and 1. This addresses the issue of features having widely varying scales, which can bias some models that are sensitive to feature magnitudes (e.g., gradient descent-based algorithms).
Apply the same scaling to validation and test sets: Crucially, *use the scaling parameters (mean and standard deviation for standardization, min and max for normalization) derived from the training set* to rescale the validation and test sets. This ensures that the validation and test sets are transformed in the same way as the training set, preventing data leakage. The model should only ever "see" the statistical properties of the training data.

Why other options are incorrect:

Option A: Random sampling *after* rescaling is irrelevant to this problem. Random sampling before splitting will not prevent data leakage. Random sampling without fixing the random seed will cause training/validation/test splits to be different on each run, leading to different results.
Option C: Rescaling the entire dataset *before* splitting introduces data leakage. The scaling parameters (mean, standard deviation, min, max) are influenced by all data points, including those in the validation and test sets. This gives the model an unfair advantage and leads to overly optimistic performance estimates during development, which do not generalize well to unseen production data.
Option D: Rescaling training, validation, and test sets *independently* introduces data leakage. The validation and test sets will be transformed using their own scaling parameters, which are different from those used for the training set. This leads to a mismatch between the data the model is trained on and the data it is evaluated on, making the validation and test performance

unreliable.
In summary: Splitting the data *before* scaling and then applying the same scaling to the validation and test set using parameters *only* derived from the training set is the best practice to prevent data leakage, address the issue of varying feature scales, and ensure reliable performance in production.
Authoritative Links:

Data Leakage: https://www.kaggle.com/dansbecker/data-leakage
Feature Scaling: https://scikit-learn.org/stable/modules/preprocessing.html
Train/Validation/Test Split: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Question: 60

Exam Heist

A Machine Learning Specialist is assigned a TensorFlow project using Amazon SageMaker for training, and needs to continue working for an extended period with no Wi-Fi access.
Which approach should the Specialist use to continue working?

- Install Python 3 and boto3 on their laptop and continue the code development using that environment.
- Download the TensorFlow Docker container used in Amazon SageMaker from GitHub to their local environment, and use the Amazon SageMaker Python SDK to test the code. ✓
- Download TensorFlow from tensorflow.org to emulate the TensorFlow kernel in the SageMaker environment.
- Download the SageMaker notebook to their local environment, then install Jupyter Notebooks on their laptop and continue the development in a local notebook.

Explanation:
The correct answer is **B. Download the TensorFlow Docker container used in Amazon SageMaker from GitHub to their local environment, and use the Amazon SageMaker Python SDK to test the code.**
Here's why:

SageMaker Environment Replication: The goal is to replicate the SageMaker environment as closely as possible to avoid discrepancies when the code is eventually deployed back to SageMaker. Downloading the TensorFlow Docker container used by SageMaker ensures you're using the same libraries, versions, and dependencies.
Docker for Consistency: Docker containers provide a consistent and isolated environment, encapsulating all the necessary software and configurations. This avoids issues caused by differing operating systems or package versions on the local machine.
SageMaker Python SDK: The SageMaker Python SDK is designed to interact with SageMaker services. Even locally, it allows you to structure your code in a way that's compatible with SageMaker's training and deployment pipelines. You can test your code locally using the SDK without actually interacting with the SageMaker service until Wi-Fi access is restored.
Why other options are less ideal:

A: Installing Python and boto3 is a good starting point but doesn't guarantee the same TensorFlow version and dependencies as the SageMaker environment. Boto3 helps you interact with AWS services, but doesn't replicate the SageMaker environment.
C: Downloading TensorFlow from tensorflow.org provides a TensorFlow installation, but might not match the specific version and configurations used in the SageMaker environment. It also doesn't address the other dependencies and environment settings.
D: Downloading the SageMaker notebook and installing Jupyter Notebooks is a good idea for editing the notebook but will require you to install the correct TensorFlow environment and all other required packages to run without access to the SageMaker Kernel or environment. It also doesn't capture the complete environment as well as the Docker container.
Authoritative Links for further research:

Amazon SageMaker: https://aws.amazon.com/sagemaker/
SageMaker Python SDK: https://sagemaker.readthedocs.io/en/stable/
Docker: https://www.docker.com/
AWS Deep Learning Containers: https://aws.amazon.com/reInvent/news/aws-deep-learning-containers-now-available/
In summary, option B is the best approach because it allows the specialist to create a local environment that closely mimics the Amazon SageMaker environment, ensuring consistency and reducing potential issues when deploying the code back to SageMaker.

Question: 61

Exam Heist

A Machine Learning Specialist is working with a large cybersecurity company that manages security events in real time for companies around the world. The cybersecurity company wants to design a solution that will allow it to use machine learning to score malicious events as anomalies on the data as it is being ingested. The company also wants be able to save the results in its data lake for later processing and analysis.
What is the MOST efficient way to accomplish these tasks?

- Ingest the data using Amazon Kinesis Data Firehose, and use Amazon Kinesis Data Analytics Random Cut Forest (RCF) for anomaly detection. Then use Kinesis Data Firehose to stream the results to Amazon S3. ✓
- Ingest the data into Apache Spark Streaming using Amazon EMR, and use Spark MLlib with k-means to perform anomaly detection. Then store the results in an Apache Hadoop Distributed File System (HDFS) using Amazon EMR with a replication factor of three as the data lake.
- Ingest the data and store it in Amazon S3. Use AWS Batch along with the AWS Deep Learning AMIs to train a k-means model using TensorFlow on the data in Amazon S3.
- Ingest the data and store it in Amazon S3. Have an AWS Glue job that is triggered on demand transform the new data. Then use the built-in Random Cut Forest (RCF) model within Amazon SageMaker to detect anomalies in the data.

Explanation:
The most efficient solution is option A, leveraging Amazon Kinesis Data Firehose and Kinesis Data Analytics with Random Cut Forest (RCF).

Real-time Anomaly Detection: The company requires real-time anomaly detection as data is ingested. Kinesis Data Analytics enables processing streaming data in real time. RCF, a built-in algorithm in Kinesis Data Analytics, is specifically designed for anomaly detection in streaming data. https://docs.aws.amazon.com/kinesisanalytics/latest/java/how-it-works-anomaly.html
Data Ingestion and Delivery: Kinesis Data Firehose efficiently ingests streaming data and delivers it to various destinations, including S3 for data lake storage. https://aws.amazon.com/kinesis/data-firehose/
Efficiency and Scalability: This solution is highly efficient because it uses managed services optimized for streaming data processing. Kinesis Data Analytics automatically scales to handle the data stream's volume.
Cost-Effectiveness: Kinesis Data Analytics and Firehose are pay-as-you-go services, optimizing cost based on actual usage.
Options B, C, and D are less efficient:

Option B: Using Apache Spark Streaming on EMR involves more operational overhead for cluster management. While Spark MLlib can perform anomaly detection, it's not as readily integrated for real-time anomaly detection on streaming data as Kinesis Data Analytics RCF. Storing in HDFS within EMR is less efficient and scalable for a data lake compared to S3.
Option C: Training a k-means model with AWS Batch and Deep Learning AMIs is suitable for batch processing and model retraining, not real-time anomaly detection. It's less efficient for the company's need to score malicious events as they're ingested.
Option D: Using AWS Glue for data transformation and SageMaker for anomaly detection introduces latency. Triggering a Glue job on demand isn't as real-time as Kinesis Data Analytics. While SageMaker's RCF is effective, it's better suited for batch anomaly detection or online learning scenarios than for direct integration with a streaming ingestion pipeline.

Question: 62

Exam Heist

A Data Scientist wants to gain real-time insights into a data stream of GZIP files.
Which solution would allow the use of SQL to query the stream with the LEAST latency?

- Amazon Kinesis Data Analytics with an AWS Lambda function to transform the data. ✓
- AWS Glue with a custom ETL script to transform the data.
- An Amazon Kinesis Client Library to transform the data and save it to an Amazon ES cluster.
- Amazon Kinesis Data Firehose to transform the data and put it into an Amazon S3 bucket.

Explanation:
The correct answer is A, Amazon Kinesis Data Analytics with an AWS Lambda function to transform the data. Here's why:
Kinesis Data Analytics is specifically designed for real-time processing and analysis of streaming data using SQL. This inherently offers low latency querying because it operates directly on the stream without the need for batch processing or data movement to other storage services for querying.
Option A uses a Lambda function to transform the GZIP files before they are ingested into Kinesis Data Analytics. Lambda functions are serverless and execute code quickly on demand, minimizing any added latency from the transformation step. This architecture facilitates the immediate availability of transformed data for SQL queries within Kinesis Data Analytics.
Option B, AWS Glue, is primarily a batch-oriented ETL (Extract, Transform, Load) service. While Glue can process data streams, it is better suited for scheduled batch jobs and incurs higher latency compared to Kinesis Data Analytics' continuous processing.

Option C, Kinesis Client Library (KCL) with Elasticsearch (ES), involves more manual coding and infrastructure management. While KCL can read from the Kinesis stream and perform transformations, the subsequent indexing and querying within Elasticsearch introduce additional latency compared to directly querying the stream with SQL using Kinesis Data Analytics.
Option D, Kinesis Data Firehose to S3, focuses on data delivery to S3 for storage and subsequent analysis. It is not designed for real-time SQL querying of the stream itself. Querying the data in S3 would require services like Athena or Redshift Spectrum, which involve loading data and therefore incur significantly higher latency.
Therefore, only Option A provides a real-time SQL query capability on a data stream with the lowest latency. It leverages the continuous processing nature of Kinesis Data Analytics along with the fast transformation speed of Lambda functions.
Supporting Links:

Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
AWS Lambda: https://aws.amazon.com/lambda/

Question: 63

Exam Heist

A retail company intends to use machine learning to categorize new products. A labeled dataset of current products was provided to the Data Science team. The dataset includes 1,200 products. The labeled dataset has 15 features for each product such as title dimensions, weight, and price. Each product is labeled as belonging to one of six categories such as books, games, electronics, and movies.
Which model should be used for categorizing new products using the provided dataset for training?

- AnXGBoost model where the objective parameter is set to multi:softmax ✓
- A deep convolutional neural network (CNN) with a softmax activation function for the last layer
- A regression forest where the number of trees is set equal to the number of product categories
- A DeepAR forecasting model based on a recurrent neural network (RNN)

Explanation:
Here's a detailed justification for why option A is the most appropriate choice, along with supporting concepts and links for further research:
The core task is a multi-class classification problem: assigning each product to one of six distinct categories. XGBoost with the `multi:softmax` objective function is specifically designed for such scenarios. XGBoost is a powerful and efficient gradient boosting algorithm known for its accuracy, speed, and ability to handle complex datasets with numerous features. The `multi:softmax` objective directly optimizes for multi-class classification by predicting the probability of each class and selecting the class with the highest probability. Given the dataset size (1,200 products) and the number of features (15), XGBoost can likely achieve good performance without excessive computational resources.
Option B, a CNN, is generally better suited for image or sequence data where spatial or temporal relationships between features are important. While CNNs can be applied to tabular data, it typically requires significant feature engineering to represent the data in a suitable format. Furthermore, CNNs often require larger datasets to train effectively, which could be a limitation with only 1,200 samples.
Option C, a Regression Forest, is primarily designed for regression tasks where the goal is to predict a continuous value, not discrete categories. While Regression Forests can be adapted for classification, it's typically done through ensemble methods or by treating each category as a separate regression problem, which might not be as efficient or accurate as a dedicated multi-class classifier like XGBoost with `multi:softmax`. Setting the number of trees equal to the number of categories is arbitrary and lacks theoretical justification.
Option D, DeepAR, is a specialized forecasting model based on recurrent neural networks (RNNs), primarily used for time series forecasting problems. It is not applicable to the product categorization problem, which involves classifying independent instances based on their features.
Therefore, XGBoost with the `multi:softmax` objective is the most appropriate model because it's a well-established, efficient, and accurate algorithm specifically designed for multi-class classification tasks, making it suitable for the given dataset and problem.
Authoritative Links for Further Research:

XGBoost Documentation: https://xgboost.readthedocs.io/en/stable/ - Provides comprehensive information on XGBoost, including the `multi:softmax` objective function.
Multi-Class Classification: https://developers.google.com/machine-learning/glossary/multi-class-classification - A glossary entry explaining multi-class classification from Google's Machine Learning Crash Course.
scikit-learn's documentation on ensemble methods: https://scikit-learn.org/stable/modules/ensemble.html - Includes details on Random Forests (which are related to Regression Forests) and their use in classification.

Question: 64

Exam Heist

A Data Scientist is working on an application that performs sentiment analysis. The validation accuracy is poor, and the Data Scientist thinks that the cause may be a rich vocabulary and a low average frequency of words in the dataset.
Which tool should be used to improve the validation accuracy?

- Amazon Comprehend syntax analysis and entity detection
- Amazon SageMaker BlazingText cbow mode
- Natural Language Toolkit (NLTK) stemming and stop word removal
- Scikit-learn term frequency-inverse document frequency (TF-IDF) vectorizer ✓

Explanation:

The problem describes a scenario where a sentiment analysis model suffers from poor validation accuracy due to a rich vocabulary and low average word frequency. This indicates that rare words might be unduly influencing the model, while common words might not be contributing enough to the sentiment classification.
Option A (Amazon Comprehend syntax analysis and entity detection) focuses on understanding sentence structure and identifying key entities. While useful for more complex NLP tasks, it doesn't directly address the issue of vocabulary size and word frequency in improving sentiment analysis accuracy.
Option B (Amazon SageMaker BlazingText cbow mode) utilizes the Continuous Bag-of-Words (CBOW) algorithm for word embeddings. While helpful for capturing semantic relationships between words, it doesn't explicitly tackle the problem of weighting words based on their frequency within the document set, which is critical when rare words are negatively impacting model performance.
Option C (Natural Language Toolkit (NLTK) stemming and stop word removal) is a good preprocessing step. Stemming reduces words to their root form, and removing stop words (like "the", "a", "is") eliminates common, non-informative words. While helpful, it doesn't address the nuances of differing importance among the remaining words after these steps. It can potentially help reduce noise and vocabulary size, but it's not as effective as TF-IDF in assigning weights based on term importance in the context of the entire dataset.
Option D (Scikit-learn TF-IDF vectorizer) is the most appropriate solution. TF-IDF (Term Frequency-Inverse Document Frequency) assigns weights to words based on how frequently they appear in a specific document (TF) and how rarely they appear across the entire collection of documents (IDF). Rare words receive a higher IDF score, thus increasing their weight, while common words receive a lower IDF score. This helps the model focus on the more discriminative terms and reduces the impact of frequently occurring but non-informative words that are *not* removed by stop word lists. This directly addresses the problem of low-frequency words unduly influencing the model and boosts the contribution of words that are more specific and useful for sentiment classification. Therefore, TF-IDF will improve the validation accuracy by highlighting important words.
For further research, see:

Scikit-learn TF-IDF: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
Understanding TF-IDF: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

Question: 65

Exam Heist

Machine Learning Specialist is building a model to predict future employment rates based on a wide range of economic factors. While exploring the data, the
Specialist notices that the magnitude of the input features vary greatly. The Specialist does not want variables with a larger magnitude to dominate the model.
What should the Specialist do to prepare the data for model training?

- Apply quantile binning to group the data into categorical bins to keep any relationships in the data by replacing the magnitude with distribution.
- Apply the Cartesian product transformation to create new combinations of fields that are independent of the magnitude.
- Apply normalization to ensure each field will have a mean of 0 and a variance of 1 to remove any significant magnitude. ✓
- Apply the orthogonal sparse bigram (OSB) transformation to apply a fixed-size sliding window to generate new features of a similar magnitude.

Explanation:

The correct answer is C: Apply normalization to ensure each field will have a mean of 0 and a variance of 1 to remove any significant magnitude. This addresses the problem of features with widely varying magnitudes dominating the model. This technique, often called standardization or Z-score normalization, transforms the data so that each feature has a mean of 0 and

a standard deviation of 1. By doing this, each feature contributes more equitably to the model, preventing features with larger scales from unduly influencing the learning process.
Option A, quantile binning, converts numerical features into categorical features. While binning can be useful in some scenarios, it doesn't directly address the issue of magnitude differences and might lose information present in the original numerical values.
Option B, the Cartesian product transformation, creates new features by combining existing features. This would drastically increase the dimensionality of the data and does not address the magnitude issue. Furthermore, it is more relevant for feature engineering related to interactions between different features rather than scaling.
Option D, the orthogonal sparse bigram (OSB) transformation, is predominantly used in natural language processing to create features from text data by looking at pairs of words. It's not applicable to handling numerical feature magnitude differences in general machine learning problems.
Normalization is a crucial preprocessing step for many machine learning algorithms, especially those sensitive to feature scaling, such as linear regression, logistic regression, support vector machines (SVMs), and neural networks. These models are designed to work best when the input features are on similar scales. Normalization ensures that the optimization process converges more efficiently and effectively. It brings all features onto a comparable scale, leading to more stable and better performing models.
https://scikit-learn.org/stable/modules/preprocessing.htmlhttps://developers.google.com/machine-learning/data-preparation/transform/normalization

Question: 66

Exam Heist

A Machine Learning Specialist must build out a process to query a dataset on Amazon S3 using Amazon Athena. The dataset contains more than 800,000 records stored as plaintext CSV files. Each record contains 200 columns and is approximately 1.5 MB in size. Most queries will span 5 to 10 columns only.
How should the Machine Learning Specialist transform the dataset to minimize query runtime?

- Convert the records to Apache Parquet format. ✓
- Convert the records to JSON format.
- Convert the records to GZIP CSV format.
- Convert the records to XML format.

Explanation:
The correct answer is A, converting the records to Apache Parquet format. Here's why:
Parquet is a columnar storage format. Unlike row-oriented formats like CSV, JSON, and XML, Parquet stores data by columns. This is a significant advantage for analytical queries that typically access only a subset of columns, as the question specifies ("Most queries will span 5 to 10 columns only"). Athena only reads the columns needed for the query, dramatically reducing I/O and query processing time.
In this scenario, converting to Parquet directly addresses the core requirement of minimizing query runtime. Columnar storage allows Athena to efficiently skip irrelevant data, resulting in faster queries when accessing only a few columns out of the total 200. The large number of records (800,000+) and the record size (1.5 MB) highlight the benefits of columnar storage, as the volume of unnecessary data read with row-oriented formats would be substantial.
Options B, C, and D are less optimal. While GZIP CSV (C) compresses the data and reduces storage costs, it still reads the entire row for each record, mitigating the performance gains of column selection. JSON (B) and XML (D) are both row-oriented and verbose, leading to larger file sizes and slower parsing speeds than Parquet. They also don't offer the columnar selection advantages.
Therefore, utilizing a columnar format like Parquet combined with the selective querying capabilities of Athena significantly optimizes query runtime, making it the best choice. Parquet also supports schema evolution and efficient encoding techniques, further boosting performance.
Further research:

Apache Parquet: https://parquet.apache.org/
Amazon Athena Performance Tuning: https://docs.aws.amazon.com/athena/latest/ug/performance-tuning.html
Columnar Storage: https://en.wikipedia.org/wiki/Column-oriented_DBMS

Question: 67

Exam Heist

A Machine Learning Specialist is developing a daily ETL workflow containing multiple ETL jobs. The workflow consists of the following processes:
* Start the workflow as soon as data is uploaded to Amazon S3.
* When all the datasets are available in Amazon S3, start an ETL job to join the uploaded datasets with multiple terabyte-sized datasets already

stored in Amazon
S3.
* Store the results of joining datasets in Amazon S3.
* If one of the jobs fails, send a notification to the Administrator.
Which configuration will meet these requirements?

- Use AWS Lambda to trigger an AWS Step Functions workflow to wait for dataset uploads to complete in Amazon S3. Use AWS Glue to join the datasets. Use an Amazon CloudWatch alarm to send an SNS notification to the Administrator in the case of a failure. ✓
- Develop the ETL workflow using AWS Lambda to start an Amazon SageMaker notebook instance. Use a lifecycle configuration script to join the datasets and persist the results in Amazon S3. Use an Amazon CloudWatch alarm to send an SNS notification to the Administrator in the case of a failure.
- Develop the ETL workflow using AWS Batch to trigger the start of ETL jobs when data is uploaded to Amazon S3. Use AWS Glue to join the datasets in Amazon S3. Use an Amazon CloudWatch alarm to send an SNS notification to the Administrator in the case of a failure.
- Use AWS Lambda to chain other Lambda functions to read and join the datasets in Amazon S3 as soon as the data is uploaded to Amazon S3. Use an Amazon CloudWatch alarm to send an SNS notification to the Administrator in the case of a failure.

Explanation:

Here's a detailed justification for why option A is the best choice, along with supporting concepts and links:
Option A provides the most robust, scalable, and manageable solution for the described ETL workflow. It leverages purpose-built AWS services designed for each stage of the process.

AWS Lambda and Step Functions: Lambda can be triggered by S3 events (object creation) to initiate the workflow. Step Functions orchestrates the ETL process, managing dependencies and state. The "wait" state in Step Functions is ideal for pausing execution until all datasets are available in S3. This provides a reliable mechanism to ensure all required data is present before the next job begins. https://docs.aws.amazon.com/step-functions/latest/dg/concepts-wait-state.html and https://docs.aws.amazon.com/lambda/latest/dg/services-s3.html
AWS Glue: Glue is designed for data integration and ETL operations. It can efficiently join terabyte-sized datasets stored in S3, leveraging its serverless, Apache Spark-based engine. It also provides data cataloging and schema discovery. https://aws.amazon.com/glue/
Amazon CloudWatch and SNS: CloudWatch monitors the ETL jobs (especially Glue) and can be configured with alarms that trigger SNS notifications upon failure. This allows administrators to be promptly alerted to issues. https://docs.aws.amazon.com/cloudwatch/latest/monitoring/AlarmThatSendsEmail.html
The other options have significant drawbacks:

Option B: Using SageMaker notebook instances for ETL is less efficient and more expensive than Glue, especially at the terabyte scale. Notebook instances are more suited for interactive data exploration and model building, not automated ETL. Lifecycle configurations are not ideal for managing complex workflow dependencies.
Option C: AWS Batch is primarily designed for batch computing and not the best fit for triggering jobs directly upon S3 object uploads. While it can be incorporated into a workflow, Lambda + Step Functions provide a more event-driven and flexible approach.
Option D: Chaining Lambda functions can become complex and difficult to manage, especially for a multi-step ETL process involving terabyte-sized datasets. Lambda functions have execution time limits and are not ideal for long-running ETL tasks or direct manipulation of very large datasets. Lambda's memory limitations also hinder it's ability to work with very large datasets.
In summary, Option A is the most appropriate choice due to its use of serverless components, robust orchestration, scalability for large datasets, and efficient error handling.

Question: 68

Exam Heist

An agency collects census information within a country to determine healthcare and social program needs by province and city. The census form collects responses for approximately 500 questions from each citizen.
Which combination of algorithms would provide the appropriate insights? (Choose two.)

- [ ] The factorization machines (FM) algorithm
- [ ] The Latent Dirichlet Allocation (LDA) algorithm
- [x] The principal component analysis (PCA) algorithm ✓
- [x] The k-means algorithm ✓
- [ ] The Random Cut Forest (RCF) algorithm

Explanation:

The task requires identifying algorithms to gain insights from a census dataset containing 500 features per citizen, aiming to determine healthcare and social program needs by province and city.
C. The principal component analysis (PCA) algorithm: PCA is a dimensionality reduction technique. With 500 features, the dataset likely suffers from high dimensionality. PCA identifies the principal components, which are linear combinations of the original features that capture the most variance in the data. By reducing the number of features, PCA simplifies the data while preserving essential information, making further analysis more efficient and interpretable. It can help identify the most important underlying factors influencing healthcare and social program needs. [https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html]
D. The k-means algorithm: K-means is a clustering algorithm. Once PCA has reduced the dimensionality, k-means can group citizens into clusters based on their remaining features (principal components). Each cluster will represent a segment of the population with similar characteristics and needs. Analyzing the characteristics of each cluster (e.g., demographic information, health conditions, social program participation) allows the agency to tailor healthcare and social programs to specific groups within each province and city. This is more efficient than trying to analyze individual citizen responses for 500 questions across millions of citizens. [https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html]
Why other options are less suitable:

A. The factorization machines (FM) algorithm: FM is primarily used for recommendation systems and predicting user preferences based on feature interactions. While it could identify feature interactions within the census data, it's not the most direct approach for initial data exploration and segmentation for healthcare and social program needs.
B. The Latent Dirichlet Allocation (LDA) algorithm: LDA is primarily used for topic modeling in text data. Since census data is typically structured and numerical, LDA isn't directly applicable.
E. The Random Cut Forest (RCF) algorithm: RCF is used for anomaly detection. While useful for identifying unusual responses, it's not the primary technique for understanding the general population's healthcare and social program needs by province and city.
Therefore, PCA followed by k-means provides a structured approach to reduce dimensionality and then segment the population, leading to actionable insights for targeted program development and resource allocation.

Question: 69

Exam Heist

A large consumer goods manufacturer has the following products on sale:
* 34 different toothpaste variants
* 48 different toothbrush variants
* 43 different mouthwash variants
The entire sales history of all these products is available in Amazon S3. Currently, the company is using custom-built autoregressive integrated moving average
(ARIMA) models to forecast demand for these products. The company wants to predict the demand for a new product that will soon be launched.
Which solution should a Machine Learning Specialist apply?

- ○ Train a custom ARIMA model to forecast demand for the new product.
- ○ Train an Amazon SageMaker DeepAR algorithm to forecast demand for the new product. ✓
- ○ Train an Amazon SageMaker k-means clustering algorithm to forecast demand for the new product.
- ○ Train a custom XGBoost model to forecast demand for the new product.

Explanation:
The most suitable solution is to train an Amazon SageMaker DeepAR algorithm. Here's why:

DeepAR for Forecasting: DeepAR is specifically designed for probabilistic forecasting of time series data. It excels at predicting future values based on historical patterns, making it ideal for demand forecasting.
Handling Multiple Time Series: DeepAR can efficiently model multiple related time series simultaneously. This is crucial because the company has sales data for various toothpaste, toothbrush, and mouthwash variants.
Cold Start Problem: The company is launching a new product with no historical sales data. DeepAR addresses this "cold start" problem by leveraging the historical data of similar existing products to make initial predictions.
Custom ARIMA Limitations: While ARIMA models are suitable for individual time series, they don't readily handle multiple related time series or cold start scenarios. Building and maintaining 125 custom ARIMA models (34+48+43) becomes complex and less scalable than a DeepAR approach. Furthermore, ARIMA does not efficiently use information across all time series to improve forecasts for new products.
XGBoost Inappropriateness: XGBoost is excellent for classification and regression tasks, but it's not inherently designed for time series forecasting. Applying XGBoost would require significant feature engineering to convert the time series data into a format suitable for XGBoost, making DeepAR a more straightforward and efficient solution.
K-means Clustering Inapplicability: k-means clustering is used for grouping data points, not for forecasting future values. It would not be suitable for the demand prediction problem.

SageMaker Benefits: Using SageMaker DeepAR simplifies model training, deployment, and management. SageMaker provides the infrastructure and tools needed to efficiently train and deploy the DeepAR model.
Therefore, DeepAR offers the best combination of accuracy, scalability, and ease of implementation for the given demand forecasting scenario, especially with the challenge of a new product launch.
Further Research:

Amazon SageMaker DeepAR: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html
Probabilistic Time Series Forecasting with Deep Learning: https://arxiv.org/abs/1704.04110

Question: 70

Exam Heist

A Machine Learning Specialist uploads a dataset to an Amazon S3 bucket protected with server-side encryption using AWS KMS.
How should the ML Specialist define the Amazon SageMaker notebook instance so it can read the same dataset from Amazon S3?

- Define security group(s) to allow all HTTP inbound/outbound traffic and assign those security group(s) to the Amazon SageMaker notebook instance.
- ¡onfigure the Amazon SageMaker notebook instance to have access to the VPC. Grant permission in the KMS key policy to the notebook's KMS role.
- Assign an IAM role to the Amazon SageMaker notebook with S3 read access to the dataset. Grant permission in the KMS key policy to that role. ✓
- Assign the same KMS key used to encrypt data in Amazon S3 to the Amazon SageMaker notebook instance.

Explanation:
The correct answer is C. Here's why:
Amazon S3 objects encrypted with KMS keys require proper authorization for access. SageMaker notebook instances, like other AWS services, need appropriate IAM roles to interact with resources like S3. Directly assigning the KMS key to the notebook instance (option D) doesn't grant permissions; KMS keys aren't directly attached to EC2 instances or notebook instances. They're used for encryption/decryption operations controlled by IAM roles.
Option A is incorrect because opening all HTTP traffic in security groups poses significant security risks. Security groups should only allow necessary traffic. Also, security groups alone don't handle KMS decryption permissions.
Option B is partially correct. VPC configuration might be needed in some scenarios, especially if S3 has VPC endpoints. However, granting permission in the KMS key policy to the *notebook's KMS role* is not quite accurate. SageMaker notebooks don't inherently have a KMS role; instead, the IAM role assigned to the notebook should be granted KMS decryption permissions.
Option C addresses the core requirement: providing the notebook instance with the necessary permissions. By assigning an IAM role to the SageMaker notebook instance with `s3:GetObject` (read) access to the specific S3 bucket and objects, you allow it to access the encrypted data. Crucially, granting permission within the KMS key policy to *that role* (the notebook's IAM role) authorizes the role to use the KMS key to decrypt the S3 objects. This aligns with the principle of least privilege – granting only the permissions necessary to perform the task. The KMS key policy must explicitly allow the notebook instance's IAM role to perform `kms:Decrypt` on the key.
In summary, accessing KMS-encrypted S3 data from a SageMaker notebook instance requires:
An IAM role attached to the SageMaker notebook instance.
The IAM role having `s3:GetObject` permissions on the S3 bucket/objects.
The KMS key policy granting the IAM role `kms:Decrypt` permission.

This approach securely grants the notebook instance the necessary permissions to access the encrypted data without compromising security.
Relevant Documentation:

IAM roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
KMS key policies: https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html
SageMaker IAM roles: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html
S3 Bucket Permissions: https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-iam.html

Question: 71

Exam Heist

A Data Scientist needs to migrate an existing on-premises ETL process to the cloud. The current process runs at regular time intervals and uses PySpark to combine and format multiple large data sources into a single consolidated output for downstream processing.
The Data Scientist has been given the following requirements to the cloud solution:
⇨ Combine multiple data sources.

- Reuse existing PySpark logic.
- Run the solution on the existing schedule.
- Minimize the number of servers that will need to be managed.

Which architecture should the Data Scientist use to build this solution?

- Write the raw data to Amazon S3. Schedule an AWS Lambda function to submit a Spark step to a persistent Amazon EMR cluster based on the existing schedule. Use the existing PySpark logic to run the ETL job on the EMR cluster. Output the results to a processed location in Amazon S3 that is accessible for downstream use.
- Write the raw data to Amazon S3. Create an AWS Glue ETL job to perform the ETL processing against the input data. Write the ETL job in PySpark to leverage the existing logic. Create a new AWS Glue trigger to trigger the ETL job based on the existing schedule. Configure the output target of the ETL job to write to a processed location in Amazon S3 that is accessible for downstream use. ✓
- Write the raw data to Amazon S3. Schedule an AWS Lambda function to run on the existing schedule and process the input data from Amazon S3. Write the Lambda logic in Python and implement the existing PySpark logic to perform the ETL process. Have the Lambda function output the results to a processed location in Amazon S3 that is accessible for downstream use.
- Use Amazon Kinesis Data Analytics to stream the input data and perform real-time SQL queries against the stream to carry out the required transformations within the stream. Deliver the output results to a processed location in Amazon S3 that is accessible for downstream use.

Explanation:

Here's a detailed justification for why option B is the best solution, along with supporting concepts and links:

Option B leverages AWS Glue, a fully managed ETL (Extract, Transform, Load) service, which directly addresses the requirement of minimizing server management. Glue allows Data Scientists to author ETL jobs using PySpark, enabling reuse of existing logic with minimal code modification. A Glue trigger can be scheduled to execute the ETL job at regular intervals, meeting the scheduling requirement. Glue natively integrates with Amazon S3, allowing it to read raw data from S3 and write processed data back to S3. Glue's serverless nature eliminates the need to provision and manage EC2 instances or EMR clusters, simplifying the operational overhead.

Option A, while utilizing EMR, involves managing a persistent cluster, contradicting the minimization of server management. Lambda functions, while serverless, have execution time limits and may not be suitable for large data processing tasks inherent in ETL. Furthermore, orchestrating Spark jobs from Lambda adds complexity.

Option C attempts to use Lambda for the entire ETL process. While Lambda is serverless, it's not designed for large-scale data transformation. PySpark is designed to run on a distributed cluster. Trying to reimplement the PySpark logic inside a single Lambda function defeats the purpose and it exceeds the execution time limitations for Lambda.

Option D utilizes Kinesis Data Analytics, which is optimized for real-time data streaming and analysis. This is not the requirement. The data is processed at specific intervals so real time analytics is not applicable.

In summary, AWS Glue provides a serverless, fully managed environment perfectly suited for running PySpark-based ETL jobs on a schedule, fulfilling all the given requirements efficiently.

Relevant links:

AWS Glue Documentation: https://aws.amazon.com/glue/
AWS Glue ETL Jobs: https://docs.aws.amazon.com/glue/latest/dg/add-job.html
AWS Glue Triggers: https://docs.aws.amazon.com/glue/latest/dg/glue-triggers.html
Amazon EMR Documentation: https://aws.amazon.com/emr/
AWS Lambda Documentation: https://aws.amazon.com/lambda/
Amazon Kinesis Data Analytics Documentation: https://aws.amazon.com/kinesis/data-analytics/

Question: 72

Exam Heist

A Data Scientist is building a model to predict customer churn using a dataset of 100 continuous numerical features. The Marketing team has not provided any insight about which features are relevant for churn prediction. The Marketing team wants to interpret the model and see the direct impact of relevant features on the model outcome. While training a logistic regression model, the Data Scientist observes that there is a wide gap between the training and validation set accuracy.

Which methods can the Data Scientist use to improve the model performance and satisfy the Marketing team's needs? (Choose two.)

- [x] Add L1 regularization to the classifier ✓
- [ ] Add features to the dataset
- [x] Perform recursive feature elimination ✓
- [ ] Perform t-distributed stochastic neighbor embedding (t-SNE)
- [ ] Perform linear discriminant analysis

Explanation:

The Data Scientist faces two main challenges: high variance (overfitting) in the logistic regression model indicated by the gap between training and validation accuracy, and the need for model interpretability to satisfy the Marketing team. L1 regularization (Option A) addresses both concerns. L1 regularization adds a penalty to the model's loss function proportional to the absolute value of the coefficients. This encourages sparsity in the model, effectively shrinking the coefficients of less important features to zero. This reduces model complexity, mitigating overfitting and improving generalization to the validation set. Because L1 regularization zeroes out unimportant features, it also performs feature selection, making the model easier to interpret as the Marketing team can focus only on the non-zero coefficients, which directly show the impact of each feature on the churn prediction.
Recursive feature elimination (RFE) (Option C) is another valid approach. RFE works by iteratively training a model, ranking features based on their importance, and removing the least important feature(s) until a desired number of features is reached. This process helps identify the most relevant features for prediction, addressing the Marketing team's need for interpretable features that have a direct impact on model outcome, while also reducing dimensionality.
Option B (Adding features) is unlikely to improve the model's performance in this case. The model is already overfitting, and adding more features would likely exacerbate the problem. Furthermore, it would make the model less interpretable. Option D (t-SNE) is a dimensionality reduction technique primarily used for visualizing high-dimensional data in lower dimensions (e.g., 2D or 3D). It's not directly helpful for improving the model's predictive performance or model interpretability. Option E (Linear Discriminant Analysis) is a dimensionality reduction technique and classification algorithm but may not be as effective as L1 regularization or RFE for feature selection and regularization, especially when dealing with a large number of features. It primarily focuses on maximizing class separability rather than sparsity.
Therefore, L1 regularization tackles both the overfitting and interpretability issues by reducing model complexity through feature selection and revealing the direct impact of the remaining features on the prediction. RFE helps identify features with impact.
Relevant Links:

L1 Regularization: https://scikit-learn.org/stable/modules/linear_model.html#lasso
Recursive Feature Elimination: https://scikit-learn.org/stable/modules/feature_selection.html#recursive-feature-elimination

Question: 73

Exam Heist

An aircraft engine manufacturing company is measuring 200 performance metrics in a time-series. Engineers want to detect critical manufacturing defects in near-real time during testing. All of the data needs to be stored for offline analysis.
What approach would be the MOST effective to perform near-real time defect detection?

- Use AWS IoT Analytics for ingestion, storage, and further analysis. Use Jupyter notebooks from within AWS IoT Analytics to carry out analysis for anomalies.
- Use Amazon S3 for ingestion, storage, and further analysis. Use an Amazon EMR cluster to carry out Apache Spark ML k-means clustering to determine anomalies.
- Use Amazon S3 for ingestion, storage, and further analysis. Use the Amazon SageMaker Random Cut Forest (RCF) algorithm to determine anomalies.
- Use Amazon Kinesis Data Firehose for ingestion and Amazon Kinesis Data Analytics Random Cut Forest (RCF) to perform anomaly detection. Use Kinesis Data Firehose to store data in Amazon S3 for further analysis. ✓

Explanation:
The correct answer is D because it outlines the most effective approach for near real-time defect detection in time-series data with subsequent storage for offline analysis. Kinesis Data Firehose excels at ingesting streaming data and delivering it to destinations like S3. Kinesis Data Analytics, specifically the Random Cut Forest (RCF) algorithm, is designed for real-time anomaly detection in streaming data. RCF is well-suited for identifying deviations in high-dimensional data, which aligns with the 200 performance metrics described in the scenario. This allows for immediate identification of critical manufacturing defects during testing. The use of Kinesis Data Firehose to simultaneously store the data in S3 enables offline analysis, fulfilling all requirements of the problem statement.
Option A is less suitable because AWS IoT Analytics, while capable of ingestion and storage, is not optimized for the near real-time anomaly detection required here. Jupyter notebooks, while versatile, are not ideal for continuous real-time analysis compared to Kinesis Data Analytics. Option B, using S3 for ingestion, lacks the necessary real-time processing capabilities. While EMR with Spark ML can perform anomaly detection, it is less suited for low-latency, continuous analysis than Kinesis Data Analytics. Option C, storing in S3 directly for ingestion is inappropriate for streaming data. SageMaker RCF is suitable for anomaly detection, but not in the direct context of real-time streaming data ingestion. It would require building custom ingestion pipeline to process the data from S3.
Key considerations include the need for real-time anomaly detection (Kinesis Data Analytics), handling of streaming data (Kinesis Data Firehose), suitability of RCF for anomaly detection (Kinesis Data Analytics), and the requirement for data storage for offline analysis (Kinesis Data Firehose -> S3).

Further Research:

Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/
Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
Random Cut Forest (RCF) Algorithm: https://docs.aws.amazon.com/sagemaker/latest/dg/anomaly-detection.html

Question: 74

Exam Heist

A Machine Learning team runs its own training algorithm on Amazon SageMaker. The training algorithm requires external assets. The team needs to submit both its own algorithm code and algorithm-specific parameters to Amazon SageMaker.
What combination of services should the team use to build a custom algorithm in Amazon SageMaker? (Choose two.)

- [ ] AWS Secrets Manager
- [ ] AWS CodeStar
- [x] Amazon ECR ✓
- [ ] Amazon ECS
- [x] Amazon S3 ✓

Explanation:
The correct answer is C and E: Amazon ECR and Amazon S3. Let's break down why.
Amazon ECR (Elastic Container Registry):

SageMaker custom algorithms often involve packaging your training code and its dependencies into Docker containers. ECR is a fully managed Docker container registry that allows you to easily store, manage, and deploy Docker container images.
The team can containerize their training algorithm code, including any custom libraries or dependencies, and then push the resulting Docker image to ECR. SageMaker can then pull this image and use it to run training jobs. This makes the training environment consistent and portable.
Using containers makes deploying custom algorithms and defining its environment repeatable and simplified.
https://aws.amazon.com/ecr/
Amazon S3 (Simple Storage Service):

S3 is an object storage service used for storing data such as training datasets, model artifacts, and in this case, algorithm-specific parameters.
The Machine Learning team can store algorithm-specific parameters, configuration files, or external assets needed by the training algorithm in S3.
SageMaker can access these parameters during the training process, enabling the algorithm to be configured as needed. S3 is a central place to store data for training, including both the training data and the metadata for the training job.
https://aws.amazon.com/s3/
Why the other options are incorrect:

A. AWS Secrets Manager: Secrets Manager is for managing secrets (e.g., database credentials, API keys), not for storing algorithm parameters or container images. While credentials could be passed to the training job via secrets manager, it's not a primary component for building a custom algorithm *itself*.
B. AWS CodeStar: CodeStar is a service for quickly developing, building, and deploying applications on AWS. It helps set up CI/CD pipelines. While useful for software development in general, it's not a direct requirement for creating a custom SageMaker algorithm. The focus of the problem is storage and execution environment of the algorithm.
D. Amazon ECS (Elastic Container Service): ECS is a container orchestration service, but for the specific task of creating and using a custom SageMaker training algorithm, ECR for image storage and S3 for parameters are more directly relevant. ECS is focused on running containerized applications, while the team here is focused on packaging a training algorithm for SageMaker.
In summary, ECR is used to store and manage the containerized algorithm code, and S3 is used to store algorithm-specific parameters. These two services, when combined, allow the Machine Learning team to build and execute their custom algorithm within SageMaker.

Question: 75

Exam Heist

A Machine Learning Specialist wants to determine the appropriate SageMakerVariantInvocationsPerInstance setting for an endpoint automatic scaling configuration. The Specialist has performed a load test on a single instance and determined that peak requests per second (RPS) without service degradation is about 20 RPS. As this is the first deployment, the Specialist intends to set the invocation safety factor to 0.5.

Based on the stated parameters and given that the invocations per instance setting is measured on a per-minute basis, what should the Specialist set as the
SageMakerVariantInvocationsPerInstance setting?

- 10
- 30
- 600 ✓
- 2,400

Explanation:

Here's a detailed justification for why the correct answer is C (600):

The question requires calculating the appropriate `SageMakerVariantInvocationsPerInstance` setting for SageMaker endpoint autoscaling, considering the safe RPS a single instance can handle. The provided information includes: peak RPS per instance (20), invocation safety factor (0.5), and the fact that the setting is measured on a per-minute basis.

First, calculate the safe RPS per instance by applying the safety factor: Safe RPS = Peak RPS *Safety Factor = 20 RPS* 0.5 = 10 RPS. This step accounts for variance and ensures endpoint stability during scaling.

Next, convert the safe RPS to invocations per minute. Since there are 60 seconds in a minute, multiply the safe RPS by 60: Invocations per Minute = Safe RPS *60 seconds/minute = 10 RPS* 60 seconds/minute = 600 invocations/minute.

Therefore, the `SageMakerVariantInvocationsPerInstance` setting should be set to 600. This means that SageMaker Auto Scaling will attempt to keep the number of invocations per instance at or below 600 per minute. If the actual invocations per instance exceed this threshold, SageMaker Auto Scaling will initiate a scale-out event (adding more instances) to distribute the load. The safety factor is crucial in preventing overload and maintaining performance. Setting it lower would result in more scaling operations and higher costs. A higher value may lead to reduced performance.

Options A, B, and D are incorrect as they do not properly account for the safety factor and/or the conversion from requests per second to requests per minute. A and B disregard converting RPS to RPM and fail to use the safety factor effectively. D incorrectly calculates the Invocations per minute.

Relevant Documentation:

SageMaker Automatic Scaling: https://docs.aws.amazon.com/sagemaker/latest/dg/endpoint-auto-scaling.html
Target Tracking Scaling Policies: https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-target-tracking.html

Question: 76

Exam Heist

A company uses a long short-term memory (LSTM) model to evaluate the risk factors of a particular energy sector. The model reviews multi-page text documents to analyze each sentence of the text and categorize it as either a potential risk or no risk. The model is not performing well, even though the Data Scientist has experimented with many different network structures and tuned the corresponding hyperparameters.
Which approach will provide the MAXIMUM performance boost?

- Initialize the words by term frequency-inverse document frequency (TF-IDF) vectors pretrained on a large collection of news articles related to the energy sector.
- Use gated recurrent units (GRUs) instead of LSTM and run the training process until the validation loss stops decreasing.
- Reduce the learning rate and run the training process until the training loss stops decreasing.
- Initialize the words by word2vec embeddings pretrained on a large collection of news articles related to the energy sector. ✓

Explanation:

The question addresses improving the performance of an LSTM model for risk factor evaluation in the energy sector, specifically analyzing text documents to classify sentences as potential risks or no risks. While the data scientist has tuned the model architecture and hyperparameters, the model's performance is still suboptimal.

The most impactful approach is initializing word embeddings with pre-trained word2vec vectors (Option D). Here's why:

Pre-trained Word Embeddings: Word2vec is a technique for learning distributed representations of words, capturing semantic relationships. By pre-training word2vec on a large corpus of energy sector news articles, the model gains a contextual understanding of relevant terminology and their interrelations *before* training on the specific risk analysis task. This provides a solid foundation.

Knowledge Transfer: The model benefits from knowledge learned from a vast dataset, transferring this knowledge to the target task. This is particularly useful when the target dataset is relatively small, as it reduces the reliance on learning word representations from scratch.

Improved Generalization: Pre-trained embeddings enhance generalization, as the model can better handle unseen words or phrases that are semantically similar to those in the pre-training corpus.

TF-IDF Limitations (Option A): TF-IDF (term frequency-inverse document frequency) measures the importance of a term in a document relative to a collection of documents. While useful for information retrieval, it doesn't capture semantic relationships between words like word2vec does. It's a bag-of-words model, not a contextual embedding.
GRU vs. LSTM (Option B): GRUs are a simplified version of LSTMs and might offer slight performance improvements in some cases, but the core issue here isn't model architecture complexity. The lack of rich word representations is a more significant bottleneck.
Learning Rate Adjustment (Option C): Reducing the learning rate can help fine-tune the model and prevent overfitting, but it addresses optimization issues rather than the fundamental issue of poor word representations. It is a step in the right direction, but a smaller gain than Option D.

Therefore, initializing with word2vec embeddings allows the model to leverage existing knowledge and understand the nuances of the energy sector's language, resulting in a significant performance boost by providing a superior starting point for the LSTM network.
Authoritative Links:

Word2Vec: https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf
Using pre-trained word embeddings: https://towardsdatascience.com/using-pre-trained-word-embeddings-in-pytorch-c253eb6ff100

Question: 77

Exam Heist

A Machine Learning Specialist needs to move and transform data in preparation for training. Some of the data needs to be processed in near-real time, and other data can be moved hourly. There are existing Amazon EMR MapReduce jobs to clean and feature engineering to perform on the data. Which of the following services can feed data to the MapReduce jobs? (Choose two.)

- [ ] AWS DMS
- [x] Amazon Kinesis ✓
- [x] AWS Data Pipeline ✓
- [ ] Amazon Athena
- [ ] Amazon ES

Explanation:
The correct answers are B and C: Amazon Kinesis and AWS Data Pipeline. Here's why:

Amazon Kinesis: Kinesis is designed for real-time data streaming. It can ingest, buffer, and process high-velocity data streams, making it suitable for feeding near-real-time data to MapReduce jobs in EMR. Kinesis Data Streams, specifically, is well-suited for handling continuous data flow. https://aws.amazon.com/kinesis/data-streams/
AWS Data Pipeline: Data Pipeline is a managed orchestration service for data movement and transformation. It allows you to create complex data workflows with dependencies, scheduling, and error handling. It can schedule hourly or other batch-oriented data movements to EMR for processing. It is ideal for regularly moving data, running scripts, or initiating EMR jobs. https://aws.amazon.com/datapipeline/
Here's why the other options are not the best fit:

AWS DMS (Database Migration Service): DMS is primarily for migrating databases to AWS or between databases. While it can move data, it's not the most efficient or appropriate choice for constantly feeding data to EMR jobs, particularly in near-real-time.
Amazon Athena: Athena is a query service that allows you to analyze data directly from S3 using SQL. It is more suitable for querying existing data lakes, not for moving data to EMR.
Amazon ES (Elasticsearch Service): Elasticsearch Service is a search and analytics engine. It's designed for indexing and searching data, not for feeding data to MapReduce jobs. It would be more relevant if the requirement was to search the processed data after it has been refined in EMR.

Question: 78

Exam Heist

A Machine Learning Specialist previously trained a logistic regression model using scikit-learn on a local machine, and the Specialist now wants to deploy it to production for inference only.
What steps should be taken to ensure Amazon SageMaker can host a model that was trained locally?

- [x] Build the Docker image with the inference code. Tag the Docker image with the registry hostname and upload it to Amazon ECR. ✓

- Serialize the trained model so the format is compressed for deployment. Tag the Docker image with the registry hostname and upload it to Amazon S3.
- Serialize the trained model so the format is compressed for deployment. Build the image and upload it to Docker Hub.
- Build the Docker image with the inference code. Configure Docker Hub and upload the image to Amazon ECR.

Explanation:

The correct answer is A. Here's a detailed justification:

To deploy a locally trained scikit-learn model on SageMaker for inference, the model needs to be packaged in a way that SageMaker can understand and execute. This involves containerizing the model and its inference code within a Docker image.

Why Docker? SageMaker uses Docker containers to provide a consistent and reproducible environment for hosting models. Docker allows you to package the model, its dependencies (like scikit-learn), and the inference script into a single, self-contained unit. This ensures the model runs the same way in SageMaker as it did locally.

Steps involved:

Build the Docker Image: The Docker image should contain:

The serialized (saved) model file (e.g., using `joblib` or `pickle`).
An inference script (e.g., `inference.py`) that loads the model and defines how to perform inference. This script will receive prediction requests and return predictions.
Any necessary dependencies specified in a requirements file (e.g., `requirements.txt`).
A web server like Flask or Gunicorn that serves prediction requests.

Tag the Docker Image: Before pushing the image, you must tag it using your AWS account ID and the region's Elastic Container Registry (ECR) hostname. This tells Docker where to push the image.

Upload to Amazon ECR: Amazon ECR is a fully managed Docker container registry that allows you to store, manage, and deploy Docker container images. By pushing the image to ECR, you make it accessible to SageMaker. SageMaker can then pull the image from ECR and use it to deploy the model endpoint.

Why other options are incorrect:

B: While serializing the model is important, uploading the image directly to S3 is not how SageMaker deploys Docker container-based models. ECR is the dedicated container registry service for AWS.

C: Docker Hub is a public container registry. While you could potentially use it, it's generally recommended to use ECR for private model deployments within AWS for security and access control reasons. Also, building an image without inference code is incorrect.

D: Configuring Docker Hub is irrelevant since the image needs to be in Amazon ECR, and you must have inference code in the docker image.

Supporting Links:

SageMaker Inference with Custom Docker Containers: https://docs.aws.amazon.com/sagemaker/latest/dg/CreateModel.html
Amazon ECR: https://aws.amazon.com/ecr/
SageMaker Scikit-learn example: https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-python-sdk/scikit_learn_inference_pipeline/scikit_learn_inference_pipeline.html

Question: 79

Exam Heist

A trucking company is collecting live image data from its fleet of trucks across the globe. The data is growing rapidly and approximately 100 GB of new data is generated every day. The company wants to explore machine learning uses cases while ensuring the data is only accessible to specific IAM users.
Which storage option provides the most processing flexibility and will allow access control with IAM?

- Use a database, such as Amazon DynamoDB, to store the images, and set the IAM policies to restrict access to only the desired IAM users.
- Use an Amazon S3-backed data lake to store the raw images, and set up the permissions using bucket policies. ✓
- Setup up Amazon EMR with Hadoop Distributed File System (HDFS) to store the files, and restrict access to the EMR instances using IAM policies.
- Configure Amazon EFS with IAM policies to make the data available to Amazon EC2 instances owned by the IAM users.

Explanation:

The correct answer is B: Use an Amazon S3-backed data lake to store the raw images, and set up the permissions using bucket policies. Here's a detailed justification:

Scalability and Cost-Effectiveness: Amazon S3 is designed for massive scalability and cost-effective storage of large amounts of unstructured data like images. It's a highly suitable choice for the 100 GB daily data ingestion. Data lakes are commonly built on S3 due to its scalability and cost efficiency for large data sets.
IAM Integration: S3 offers robust integration with IAM, enabling granular access control. You can define bucket policies and IAM policies that specify which IAM users or roles have permissions to access, read, write, or delete objects within the S3 bucket. This directly addresses the requirement of restricting data access to specific IAM users.
Processing Flexibility: Storing the data in its raw format within S3 provides maximum flexibility for future machine learning use cases. You can readily process the data using various AWS services like AWS Glue, Amazon SageMaker, Amazon EMR, or AWS Lambda.
Data Lake Architecture: Building a data lake on S3 allows you to store data in its raw format without requiring upfront transformation, which aligns with the company's need to explore machine learning use cases before fully defining the processing pipeline.
DynamoDB (Option A): While DynamoDB supports IAM policies, storing images directly in DynamoDB can be less cost-effective and may require more complex data modeling compared to storing them as objects in S3. DynamoDB is better suited for structured data.
Amazon EMR with HDFS (Option C): Setting up Amazon EMR solely for storage using HDFS is overkill for this scenario. EMR is designed for large-scale data processing, not primarily for storage. HDFS can be more complex to manage than S3.
Amazon EFS (Option D): Amazon EFS is a file system designed for shared access by multiple EC2 instances. While EFS supports IAM policies, it is more appropriate when you need file-system semantics and shared access from multiple compute instances. Using EFS solely for storage can be less cost-effective than S3, especially for data that is accessed less frequently. Therefore, using an S3-backed data lake offers the best balance of scalability, cost-effectiveness, IAM integration, and processing flexibility for storing and accessing the trucking company's image data.
Authoritative links:

Amazon S3 Documentation
IAM Policies for S3
AWS Data Lake

Question: 80

Exam Heist

A credit card company wants to build a credit scoring model to help predict whether a new credit card applicant will default on a credit card payment. The company has collected data from a large number of sources with thousands of raw attributes. Early experiments to train a classification model revealed that many attributes are highly correlated, the large number of features slows down the training speed significantly, and that there are some overfitting issues.
The Data Scientist on this project would like to speed up the model training time without losing a lot of information from the original dataset.
Which feature engineering technique should the Data Scientist use to meet the objectives?

- Run self-correlation on all features and remove highly correlated features
- Normalize all numerical values to be between 0 and 1
- Use an autoencoder or principal component analysis (PCA) to replace original features with new features ✓
- Cluster raw data using k-means and use sample data from each cluster to build a new dataset

Explanation:
The correct answer is **C. Use an autoencoder or principal component analysis (PCA) to replace original features with new features.**
Here's why:
The problem describes a dataset with high dimensionality, high correlation between features, slow training times, and overfitting. These issues are classic scenarios where dimensionality reduction techniques are beneficial.

PCA and Autoencoders for Dimensionality Reduction: PCA and autoencoders are feature extraction methods specifically designed to reduce the number of features while retaining as much variance (information) as possible. PCA achieves this through linear transformations, creating uncorrelated principal components. Autoencoders, particularly non-linear ones, can capture more complex relationships and potentially represent data more compactly.
Addressing the Problems: These techniques directly address the stated issues:

Reduced Training Time: Fewer features means faster model training.
Correlation Handling: Both PCA and autoencoders aim to create new, less correlated features. PCA guarantees uncorrelated features, while autoencoders learn compressed representations that ideally remove redundancies.
Overfitting Mitigation: By reducing the dimensionality, the model has fewer parameters to fit, decreasing the risk of overfitting, especially when dealing with a limited dataset size relative to the number of features.
Let's analyze the other options:

A. Run self-correlation on all features and remove highly correlated features: This is a form of feature selection. While removing highly correlated features is helpful, it doesn't necessarily address the high dimensionality as effectively as PCA or autoencoders. It can also lead to information loss if you're just removing features without capturing their underlying variance into new features.
B. Normalize all numerical values to be between 0 and 1: Normalization is a good preprocessing step for many machine learning algorithms, especially gradient descent-based ones, as it helps with faster convergence. However, it doesn't reduce the number of features or directly address the correlation issue. It primarily helps improve the performance of certain algorithms on the existing feature set, but doesn't speed up training due to feature reduction or prevent overfitting due to fewer features.
Why C is superior: PCA and autoencoders perform feature *extraction*, creating a new, lower-dimensional feature space that captures the essence of the original data. This is more effective at reducing dimensionality and mitigating the issues described than simply selecting a subset of the original features.
In summary, PCA and autoencoders are the most appropriate feature engineering techniques to address high dimensionality, high correlation, slow training, and overfitting issues in the context of this credit card scoring model.
Authoritative Links:

PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Autoencoders: https://www.tensorflow.org/tutorials/generative/autoencoder

Question: 81

Exam Heist

A Data Scientist is training a multilayer perception (MLP) on a dataset with multiple classes. The target class of interest is unique compared to the other classes within the dataset, but it does not achieve and acceptable recall metric. The Data Scientist has already tried varying the number and size of the MLP's hidden layers, which has not significantly improved the results. A solution to improve recall must be implemented as quickly as possible.
Which techniques should be used to meet these requirements?

- Gather more data using Amazon Mechanical Turk and then retrain
- Train an anomaly detection model instead of an MLP
- Train an XGBoost model instead of an MLP
- Add class weights to the MLP's loss function and then retrain ✓

Explanation:
The scenario describes a class imbalance problem where the target class has low recall despite efforts to optimize the model architecture. The fastest and most effective way to address this in the context of an existing MLP is to adjust the loss function to account for the imbalance.
Option D, adding class weights to the MLP's loss function and retraining, directly addresses the problem. By assigning higher weights to the under-represented class (the target class with low recall), the model is penalized more for misclassifying instances of that class. This encourages the model to prioritize correctly classifying the target class, thus improving recall. This approach is a simple modification to the training process and doesn't require significant changes to the model architecture or data collection.
Option A, gathering more data using Amazon Mechanical Turk, might improve the model's performance eventually, but it's a time-consuming process. Gathering, validating, and cleaning the new data takes considerable effort, making it a slow solution.
Option B, training an anomaly detection model, is inappropriate. Anomaly detection is designed for identifying unusual data points in a single class or without labels, while the problem deals with supervised multi-class classification.
Option C, training an XGBoost model, could potentially improve performance, but it involves retraining a completely different model, which takes time and resources, including hyperparameter tuning and evaluation. While XGBoost is a powerful algorithm, adding class weights is a faster initial approach to see if improved recall can be achieved with the existing model.
Therefore, modifying the MLP's loss function to incorporate class weights is the quickest way to potentially improve the recall of the under-represented class.
https://developers.google.com/machine-learning/crash-course/improving-neural-nets/balancing-training-datahttps://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

Question: 82

Exam Heist

A Machine Learning Specialist works for a credit card processing company and needs to predict which transactions may be fraudulent in near-real time.
Specifically, the Specialist must train a model that returns the probability that a given transaction may fraudulent.
How should the Specialist frame this business problem?

- Streaming classification
- Binary classification ✓
- Multi-category classification
- Regression classification

Explanation:

The correct answer is **B. Binary classification.** Here's a detailed justification:

The problem requires predicting whether a transaction is fraudulent or not. This translates to two possible outcomes: fraudulent (positive class) or not fraudulent (negative class). This inherently defines a binary classification problem. Binary classification deals with classifying data points into one of two distinct categories.

Option A, Streaming classification, refers to the method of applying a classification model to a continuous stream of data in real-time or near real-time. While the specialist needs to make predictions in near-real-time, the *type* of problem isn't determined by the streaming aspect. It is still fundamentally a binary classification problem, even if the data comes in a stream. Streaming classification is a deployment consideration, not the problem framing itself.

Option C, Multi-category classification, involves classifying data into more than two categories. This doesn't fit the scenario because there are only two outcomes: fraudulent or not fraudulent.

Option D, Regression classification is incorrect. Regression aims to predict a continuous numerical value (e.g., house price, temperature). It's not appropriate for predicting a categorical outcome like fraud. The problem asks for the *probability* of a transaction being fraudulent, but that probability is still used to classify the transaction into one of two categories. The model will use probability to classify the transaction, but the business problem is classification.

Therefore, framing the problem as binary classification is the most accurate and appropriate approach. The specialist needs to train a model that can distinguish between these two classes with high accuracy and recall, considering the cost associated with false negatives (missing fraudulent transactions).

For further reading on binary classification and model evaluation metrics (precision, recall, F1-score, AUC-ROC) relevant to this scenario, you can refer to these resources:

AWS Machine Learning Documentation: https://docs.aws.amazon.com/machine-learning/latest/dg/types-of-ml-problems.html
Binary Classification Metrics (scikit-learn): https://scikit-learn.org/stable/modules/model_evaluation.html

Question: 83

Exam Heist

A real estate company wants to create a machine learning model for predicting housing prices based on a historical dataset. The dataset contains 32 features.
Which model will meet the business requirement?

- Logistic regression
- Linear regression ✓
- K-means
- Principal component analysis (PCA)

Explanation:

Here's a detailed justification for why Linear Regression is the best choice for predicting housing prices in this scenario:

The core task is to predict a continuous numerical value (housing price) based on multiple input features (32 in this case). This makes it a regression problem. Linear Regression is a fundamental and widely used algorithm specifically designed for solving regression problems. It models the relationship between the independent variables (features) and the dependent variable (housing price) as a linear equation. The model learns the coefficients for each feature to minimize the difference between predicted and actual prices.

Option A, Logistic Regression, is primarily used for classification problems (predicting categorical outcomes), not regression. Option C, K-means, is a clustering algorithm used to group data points into clusters based on similarity. It's unsuitable for predicting a specific target variable. Option D, Principal Component Analysis (PCA), is a dimensionality reduction technique used to reduce the number of features while preserving variance. While PCA might be useful for pre-processing the data to reduce the number of features (if multicollinearity is a problem), it doesn't directly predict housing prices.

Linear regression offers simplicity and interpretability. It's relatively easy to understand how each feature contributes to the predicted price based on the learned coefficients. With 32 features, the linear relationship can be reasonably modeled by linear regression before exploring more complex models.

Therefore, Linear Regression is the most appropriate choice for this business requirement because it directly addresses the need to predict a continuous numerical value (housing price) based on multiple input features, unlike the other options which are designed for different types of machine learning tasks.

Further Research:

Linear Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
Regression Analysis: https://www.ibm.com/topics/regression-analysis

Question: 84

Exam Heist

A Machine Learning Specialist is applying a linear least squares regression model to a dataset with 1,000 records and 50 features. Prior to training, the ML
Specialist notices that two features are perfectly linearly dependent.
Why could this be an issue for the linear least squares regression model?

- ○ It could cause the backpropagation algorithm to fail during training
- ○ It could create a singular matrix during optimization, which fails to define a unique solution ✓
- ○ It could modify the loss function during optimization, causing it to fail during training
- ○ It could introduce non-linear dependencies within the data, which could invalidate the linear assumptions of the model

Explanation:

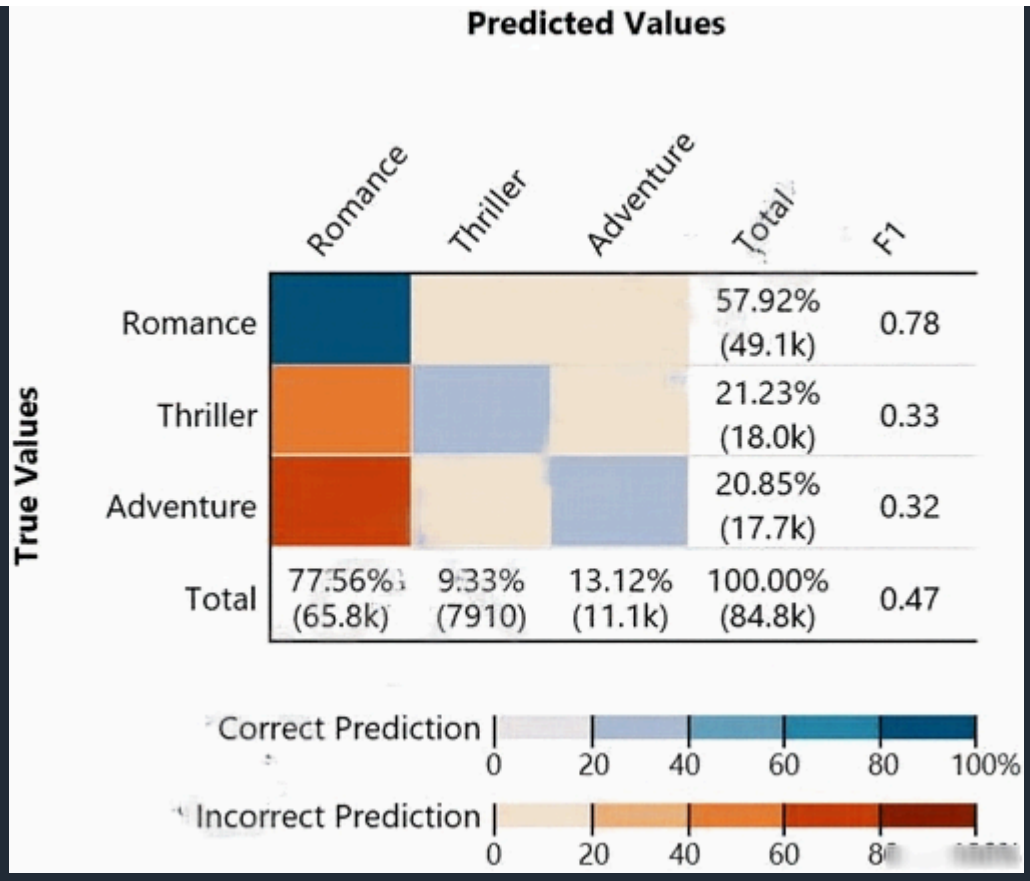
Here's a detailed justification for why option B is the correct answer, along with supporting concepts and links:
The core issue with perfectly linearly dependent features (multicollinearity) in linear least squares regression stems from its impact on the underlying mathematical operations used to find the optimal model parameters (coefficients). Linear least squares regression aims to minimize the sum of squared differences between the predicted and actual values. This minimization process typically involves solving a system of linear equations derived from the data.
When two features are perfectly linearly dependent, it means one feature can be expressed as a linear combination of the other. This redundancy introduces a problem in the matrix algebra involved in solving for the regression coefficients. Specifically, the matrix representing the features (often denoted as the design matrix or X) becomes singular or near-singular. A singular matrix is a square matrix that does not have an inverse. In the context of linear regression, the inverse of the matrix (X^TX) is required to calculate the coefficients. If X^TX is singular, its inverse doesn't exist. This means there are infinitely many possible solutions for the regression coefficients, and the algorithm cannot determine a unique, optimal solution. The model becomes unstable and sensitive to minor changes in the data. The algorithm might also crash or produce nonsensical results.
Options A, C, and D are incorrect because:

A. It could cause the backpropagation algorithm to fail during training: Backpropagation is primarily used in neural networks, not linear least squares regression. Linear regression is typically solved using direct methods (e.g., normal equations) or iterative methods like gradient descent without backpropagation.
C. It could modify the loss function during optimization, causing it to fail during training: While multicollinearity can affect the optimization process, it doesn't directly modify the loss function itself. The loss function (e.g., mean squared error) remains the same, but the optimization struggles to find the minimum due to the non-uniqueness of the solution.
D. It could introduce non-linear dependencies within the data, which could invalidate the linear assumptions of the model: Linear dependencies between features don't introduce non-linear dependencies. They simply create redundancy in the linear relationships already present. The problem is not that the linear assumption is invalidated, but that the model becomes mathematically unstable due to the singularity issue.
Authoritative Links:
Wikipedia - Multicollinearity: Provides a general overview of multicollinearity and its impact on regression analysis. https://en.wikipedia.org/wiki/Multicollinearity
StatQuest - Multicollinearity: Presents a clear explanation of multicollinearity and its effects. https://statquest.org/video/multicollinearity/
PennState - STAT 501: Regression Methods - Lesson 7: Multicollinearity: Details about multicollinearity, its causes, and ways to detect and address it. https://online.stat.psu.edu/stat501/lesson/7

Question: 85

Exam Heist

Given the following confusion matrix for a movie classification model, what is the true class frequency for Romance and the predicted class frequency for
Adventure?



- The true class frequency for Romance is 77.56% and the predicted class frequency for Adventure is 20.85%
- The true class frequency for Romance is 57.92% and the predicted class frequency for Adventure is 13.12% ✓
- The true class frequency for Romance is 0.78 and the predicted class frequency for Adventure is (0.47-0.32)
- The true class frequency for Romance is 77.56% $\frac{}{\text{True}}$ 0.78 and the predicted class frequency for Adventure is 20.85% $\frac{}{\text{True}}$ 0.32

Explanation:

B. The true class frequency for Romance is 57.92% and the predicted class frequency for Adventure is 13.12%.
The true class frequency represents the percentage of actual Romance movies in the dataset, while the predicted class frequency indicates the percentage of movies predicted to be Adventure. These values are derived from the confusion matrix used in classification models.
Would you like a deeper explanation of how confusion matrices work in machine learning?

Question: 86

Exam Heist

A Machine Learning Specialist wants to bring a custom algorithm to Amazon SageMaker. The Specialist implements the algorithm in a Docker container supported by Amazon SageMaker.
How should the Specialist package the Docker container so that Amazon SageMaker can launch the training correctly?

- Modify the bash_profile file in the container and add a bash command to start the training program
- Use CMD config in the Dockerfile to add the training program as a CMD of the image
- Configure the training program as an ENTRYPOINT named train ✓
- Copy the training program to directory /opt/ml/train

Show Explanation

Question: 87

Exam Heist

A Data Scientist needs to analyze employment data. The dataset contains approximately 10 million observations on people across 10 different features. During the preliminary analysis, the Data Scientist notices that income and age distributions are not normal. While income levels shows a right skew as expected, with fewer individuals having a higher income, the age distribution also shows a right skew, with fewer older individuals participating in the workforce.
Which feature transformations can the Data Scientist apply to fix the incorrectly skewed data? (Choose two.)

- [ ] Cross-validation
- [x] Numerical value binning ✓
- [ ] High-degree polynomial transformation
- [x] Logarithmic transformation ✓
- [ ] One hot encoding

Explanation:
The question is asking about appropriate feature transformation techniques to address right-skewed distributions in numerical features (income and age).
Option B, Numerical value binning, is a suitable technique for handling skewed data. Binning involves grouping data into intervals or bins. This transformation can convert a skewed numerical feature into a categorical one, mitigating the impact of outliers and non-normal distributions by representing data in a more discrete manner. Specifically, it can reduce the impact of a long tail, as the outliers in the tail will be grouped together. This is because the data is no longer represented by individual values but rather by the bins they fall into.
Option D, Logarithmic transformation, is also a helpful approach for addressing right-skewed data. The logarithmic transformation compresses the range of values, reducing the impact of large values and making the distribution more symmetric or normal-like. Applying a log transformation can normalize the distribution, making it easier for machine learning models to learn from the data. Specifically, by reducing the effect of outliers and bringing values closer together. A log transformation is often used on income data, which is often right skewed.
Option A, Cross-validation, is a model evaluation technique, not a feature transformation. It helps assess model performance by splitting the data into training and validation sets.
Option C, High-degree polynomial transformation, is a technique for capturing non-linear relationships between features and the target variable. Applying it directly to skewed features won't necessarily fix the skewness and might instead lead to overfitting if not handled carefully. While sometimes it can reduce the impact of skewness on the model performance, it does not directly transform the variable in the way that a log or binning transformation does.
Option E, One-hot encoding, is a technique for transforming categorical features into numerical representations suitable for machine learning models. It is not applicable for handling skewed numerical data.
Therefore, the correct answers are B and D.
Relevant resources:

Feature Engineering: https://scikit-learn.org/stable/modules/preprocessing.html
Data Transformation: https://developers.google.com/machine-learning/data-prep/transform/normalization

Question: 88

Exam Heist

A web-based company wants to improve its conversion rate on its landing page. Using a large historical dataset of customer visits, the company has repeatedly trained a multi-class deep learning network algorithm on Amazon SageMaker. However, there is an overfitting problem: training data shows 90% accuracy in predictions, while test data shows 70% accuracy only.
The company needs to boost the generalization of its model before deploying it into production to maximize conversions of visits to purchases.
Which action is recommended to provide the HIGHEST accuracy model for the company's test and validation data?

- Increase the randomization of training data in the mini-batches used in training
- Allocate a higher proportion of the overall data to the training dataset
- Apply L1 or L2 regularization and dropouts to the training ✓
- Reduce the number of layers and units (or neurons) from the deep learning network

Explanation:
Here's a detailed justification for why option C is the most appropriate solution to the overfitting problem in the given scenario:
The problem clearly states that the deep learning model is overfitting, meaning it performs very well on the training data but poorly on unseen data (test data). This suggests the model has learned the training data too well, including its noise and specific patterns, rather than generalizing to the underlying relationship between input features and the target variable (conversion).
Option A, increasing the randomization of training data, might help slightly, but it primarily addresses issues with biased data presentation or poor data shuffling, not the core overfitting problem. It doesn't directly prevent the model from memorizing the training data.
Option B, allocating a higher proportion of data to the training set, is generally *not* a good idea when overfitting. Overfitting means the model is already too good at fitting the *existing* training data. Adding more training data *without* addressing the overfitting mechanisms will likely exacerbate the problem, leading to even worse performance on the test/validation data. More data can *sometimes* help, but *only* if it's substantially different from the existing data and forces the model to learn more robust representations. In this case, it is more of the same.
Option C, applying L1 or L2 regularization and dropouts, directly addresses overfitting. L1 and L2 regularization add a penalty to the loss function based on the magnitude of the model's weights. This encourages the model to learn smaller weights, which simplifies the model and prevents it from relying too heavily on any single feature or combination of features. Dropout randomly deactivates neurons during training, which forces the remaining neurons to learn more robust features and reduces co-adaptation between neurons. Both L1/L2 regularization and dropout help the model to generalize better to unseen data. This approach is the best way to reduce overfitting.

Option D, reducing the number of layers and units, is a valid approach to reduce overfitting, but it's more drastic than using regularization and dropout. It might work, but it is not guaranteed to yield the best results for test/validation data, and can lead to underfitting. Using the dropout and regularization keeps more of the complexity and can better find complex patterns in the data. It's best to first try regularization and dropout before drastically reducing the model's capacity.
Therefore, applying L1 or L2 regularization and dropouts (Option C) is the most targeted and appropriate action to improve the model's generalization performance, reduce overfitting, and ultimately provide the highest accuracy model for test and validation data, thus maximizing conversions on the landing page.
Relevant Links for further research:

Regularization: https://www.tensorflow.org/api_docs/python/tf/keras/regularizers
Dropout: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout
Overfitting and Underfitting: https://www.ibm.com/cloud/learn/overfitting

Question: 89

Exam Heist

A Machine Learning Specialist is given a structured dataset on the shopping habits of a company's customer base. The dataset contains thousands of columns of data and hundreds of numerical columns for each customer. The Specialist wants to identify whether there are natural groupings for these columns across all customers and visualize the results as quickly as possible.
What approach should the Specialist take to accomplish these tasks?

- Embed the numerical features using the t-distributed stochastic neighbor embedding (t-SNE) algorithm and create a scatter plot. ✓
- Run k-means using the Euclidean distance measure for different values of k and create an elbow plot.
- Embed the numerical features using the t-distributed stochastic neighbor embedding (t-SNE) algorithm and create a line graph.
- Run k-means using the Euclidean distance measure for different values of k and create box plots for each numerical column within each cluster.

Explanation:
Here's a detailed justification for why option A is the best approach, considering the problem's requirements of quickly identifying and visualizing natural groupings within numerous numerical columns across customers:
The problem requires identifying natural groupings within a high-dimensional numerical dataset and visualizing them. t-SNE (t-distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique particularly well-suited for visualizing high-dimensional data in lower dimensions (typically 2D or 3D). It excels at preserving the local structure of the data, meaning that data points that are close together in the high-dimensional space are also likely to be close together in the lower-dimensional representation. This makes it ideal for revealing clusters. Creating a scatter plot of the t-SNE reduced data will directly allow the Machine Learning Specialist to visually identify potential customer groupings or patterns. The emphasis on speed is a key factor; t-SNE, while computationally intensive, is generally faster than other clustering methods when an initial, quick visual assessment of cluster existence is needed.
K-means (Option B and D), on the other hand, requires pre-defining the number of clusters (k). While an elbow plot can help determine a suitable value for k, this process still involves multiple k-means runs and analysis, which takes more time. Moreover, k-means directly reveals cluster assignments but lacks the inherent visualization capabilities of t-SNE. While box plots (Option D) within each cluster can give insight, this increases the time to interpret the data as it's not a single visualization technique. A line graph (Option C) of t-SNE outputs is simply not a suitable visualization as it requires the data to have an inherent ordering, that t-SNE does not provide. The goal isn't to see how t-SNE changes over time, but to use t-SNE to produce the points necessary for cluster visualization using a scatter plot.
Therefore, using t-SNE and visualizing the output with a scatter plot gives the fastest way to meet the requirements.
Relevant Links for further research:

t-SNE: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
Dimensionality Reduction: https://scikit-learn.org/stable/modules/manifold.html

Question: 90

Exam Heist

A Machine Learning Specialist is planning to create a long-running Amazon EMR cluster. The EMR cluster will have 1 master node, 10 core nodes, and 20 task nodes. To save on costs, the Specialist will use Spot Instances in the EMR cluster.
Which nodes should the Specialist launch on Spot Instances?

- Master node
- Any of the core nodes
- Any of the task nodes ✓
- Both core and task nodes

Explanation:

The correct answer is C: Any of the task nodes. Here's why:

Spot Instances offer significant cost savings compared to On-Demand Instances by utilizing spare EC2 capacity. However, Spot Instances can be interrupted with little or no warning if the spot price exceeds your bid. This makes them ideal for fault-tolerant workloads.

Master nodes in an EMR cluster are crucial for cluster management and coordination. Losing a master node can lead to the entire cluster failing or becoming unstable. Therefore, it's not recommended to use Spot Instances for the master node.

Core nodes are responsible for storing data using HDFS and performing computations. While some level of data loss might be tolerable, frequent interruptions of core nodes can severely impact performance and data reliability. Using On-Demand or Reserved Instances for core nodes is generally the preferred approach.

Task nodes are primarily used for executing tasks and can be easily replaced if interrupted. Because they generally do not store persistent data or hold critical roles, the loss of a few task nodes does not severely impact cluster stability. Spot Instances are perfectly suited for this. The ephemeral nature of task nodes aligns well with the interruptible nature of Spot Instances, allowing for significant cost savings without jeopardizing the overall integrity of the cluster.

Therefore, launching task nodes on Spot Instances is the most cost-effective and reliable option in this scenario.

Further Research:

AWS EMR Pricing: https://aws.amazon.com/emr/pricing/

Using Spot Instances with EMR: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-instance-purchasing-options.html

Spot Instances Documentation: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

Question: 91

Exam Heist

A manufacturer of car engines collects data from cars as they are being driven. The data collected includes timestamp, engine temperature, rotations per minute
(RPM), and other sensor readings. The company wants to predict when an engine is going to have a problem, so it can notify drivers in advance to get engine maintenance. The engine data is loaded into a data lake for training.
Which is the MOST suitable predictive model that can be deployed into production?

- Add labels over time to indicate which engine faults occur at what time in the future to turn this into a supervised learning problem. Use a recurrent neural network (RNN) to train the model to recognize when an engine might need maintenance for a certain fault. ✓
- This data requires an unsupervised learning algorithm. Use Amazon SageMaker k-means to cluster the data.
- Add labels over time to indicate which engine faults occur at what time in the future to turn this into a supervised learning problem. Use a convolutional neural network (CNN) to train the model to recognize when an engine might need maintenance for a certain fault.
- This data is already formulated as a time series. Use Amazon SageMaker seq2seq to model the time series.

Explanation:

Here's a detailed justification for why option A is the most suitable answer, incorporating relevant cloud computing and machine learning concepts:

The problem is framed as predicting future engine faults based on historical sensor data. This inherently involves analyzing temporal sequences of data (time series). The key is to transform this into a supervised learning task, which option A correctly addresses. By adding labels indicating future fault occurrences at specific times, we provide the model with examples of what leads to failures. This allows the model to learn the patterns in the data that precede engine problems.

A Recurrent Neural Network (RNN) is particularly well-suited for time-series data. RNNs are designed to handle sequential information, maintaining a hidden state that captures dependencies across time steps. This makes them ideal for learning temporal relationships within the engine data. They can recognize patterns of RPM fluctuations, temperature changes, and other sensor readings that, over time, signal an impending fault. Specifically, Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) variants of RNNs are commonly employed to address vanishing gradient problems and capture long-range dependencies in the time series data.

Option B is incorrect because unsupervised learning (k-means) wouldn't directly predict future faults. While clustering might reveal groupings of engine states, it wouldn't establish a predictive relationship between these clusters and future failures without labelled fault data.

Option C is less appropriate. Convolutional Neural Networks (CNNs) are primarily designed for spatial data (like images). While a CNN could potentially be applied after transforming the time-series data into a suitable image-like representation (e.g., a spectrogram), this adds unnecessary complexity. RNNs are a more natural fit for time series.
Option D, using SageMaker's seq2seq, is also less optimal. Seq2seq models are usually employed when mapping one sequence to *another* sequence, like machine translation. While potentially applicable, it's an overkill for this problem. The primary goal isn't transforming the time series, but rather predicting a binary outcome (fault or no fault) at a future time based on the input sequence. An RNN configured for time-series classification is more straightforward and efficient.
In summary, labeling the data to indicate future faults transforms the problem into a supervised time-series classification task. RNNs, especially LSTMs or GRUs, are the most appropriate model for this type of problem due to their ability to learn temporal dependencies and predict future outcomes based on past sequences of data. Deploying this model with Amazon SageMaker allows for scalable training and inference in a cloud environment.
Authoritative Links:

Amazon SageMaker: https://aws.amazon.com/sagemaker/
Recurrent Neural Networks: https://deeplearning.net/tutorial/rnnlstm.html
Time Series Analysis: https://otexts.com/fpp2/

Question: 92

Exam Heist

A company wants to predict the sale prices of houses based on available historical sales data. The target variable in the company's dataset is the sale price. The features include parameters such as the lot size, living area measurements, non-living area measurements, number of bedrooms, number of bathrooms, year built, and postal code. The company wants to use multi-variable linear regression to predict house sale prices.
Which step should a machine learning specialist take to remove features that are irrelevant for the analysis and reduce the model's complexity?

- ○ Plot a histogram of the features and compute their standard deviation. Remove features with high variance.
- ○ Plot a histogram of the features and compute their standard deviation. Remove features with low variance.
- ○ Build a heatmap showing the correlation of the dataset against itself. Remove features with low mutual correlation scores.
- ○ Run a correlation check of all features against the target variable. Remove features with low target variable correlation scores. ✓

Explanation:
The correct answer is D. Here's a detailed justification:
The goal is to reduce model complexity and identify irrelevant features for predicting house sale prices using multi-variable linear regression. This is feature selection.
Option A and B focus on feature variance. While high variance can indicate a feature might be informative, and low variance might suggest a feature is essentially constant and therefore unhelpful, variance alone doesn't indicate relevance to the target variable (sale price). A feature could have high variance but no impact on sale price, or low variance but still be predictive.
Option C involves feature-feature correlation. While useful for identifying multicollinearity (highly correlated independent variables), removing features based solely on their correlation with other features doesn't guarantee that the removed feature is irrelevant to the *target variable* (sale price). Addressing multicollinearity primarily focuses on improving model interpretability and stability, not necessarily feature relevance to prediction.
Option D directly addresses feature relevance. By correlating each feature with the target variable (sale price), the machine learning specialist can identify features with a weak or nonexistent relationship with the target. Low correlation with the target indicates that the feature is likely a poor predictor of sale price. Removing such features simplifies the model, reduces noise, and potentially improves model generalization performance by focusing on the features that actually influence the predicted outcome. Linear regression relies on the correlation between independent and dependent variables; features with low correlation contribute minimal predictive power.
Therefore, assessing feature-target correlation and removing features with low scores is the most direct and effective method for removing irrelevant features and reducing model complexity in this scenario. This approach ensures that the selected features are the most pertinent for predicting sale price.
Further reading:

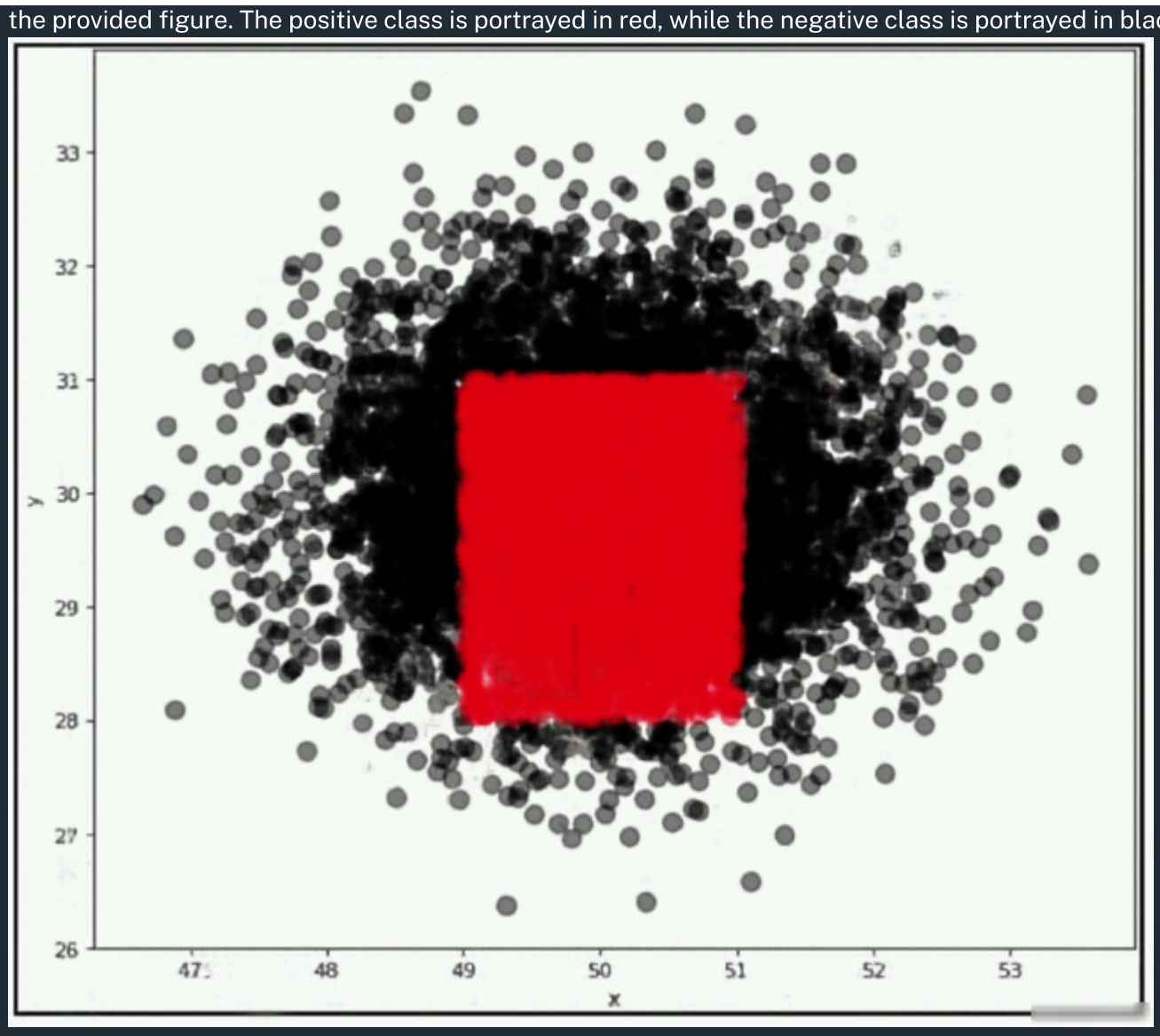
Feature Selection Techniques: https://www.analyticsvidhya.com/blog/2022/02/feature-selection-techniques-in-machine-learning/
Correlation and Linear Regression: https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Question: 93

Exam Heist

A company wants to classify user behavior as either fraudulent or normal. Based on internal research, a machine learning specialist will build a binary classifier based on two features: age of account, denoted by x, and transaction month, denoted by y. The class distributions are illustrated in

the provided figure. The positive class is portrayed in red, while the negative class is portrayed in black.



Which model would have the HIGHEST accuracy?

- Linear support vector machine (SVM)
- Decision tree ✓
- Support vector machine (SVM) with a radial basis function kernel
- Single perceptron with a Tanh activation function

Explanation:

1. A Decision tree produces a stepwise boundary that consists of rectilinear splits in the feature space that are perpendicular to the axes. The boundary is determined by the conditions specified in the tree.Support Vector Machines (SVM) produce a non-linear boundary in the feature space that separates the classes. The boundary is defined as the maximum margin hyperplane, which is the line that maximally separates the classes while having the greatest margin between the classes. The boundary can be linear, polynomial, radial basis function (RBF) or other types of non-linear functions, depending on the choice of kernel and the configuration of the SVM.
2. This answer is decision tree due to the Square decision boundaries.

Question: 94

Exam Heist

A health care company is planning to use neural networks to classify their X-ray images into normal and abnormal classes. The labeled data is divided into a training set of 1,000 images and a test set of 200 images. The initial training of a neural network model with 50 hidden layers yielded 99% accuracy on the training set, but only 55% accuracy on the test set.
What changes should the Specialist consider to solve this issue? (Choose three.)

- [ ] Choose a higher number of layers
- [x] Choose a lower number of layers ✓
- [ ] Choose a smaller learning rate
- [x] Enable dropout ✓
- [ ] Include all the images from the test set in the training set
- [x] Enable early stopping ✓

Explanation:

The situation described points to a classic case of overfitting. The model has memorized the training data instead of learning generalizable patterns, leading to high training accuracy but poor performance on unseen data (the test set). The chosen options, B, D, and F, directly address overfitting.

B. **Choose a lower number of layers:** A model with 50 hidden layers is likely too complex for a relatively small dataset of 1,000 training images. A simpler model with fewer layers reduces the model's capacity to memorize the training data, promoting better generalization. This reduction in complexity helps prevent the model from fitting the noise in the training data. https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/complexity-and-generalization

D. **Enable dropout:** Dropout is a regularization technique that randomly disables neurons during training. This prevents neurons from becoming overly reliant on specific features in the training data, forcing the network to learn more robust and generalizable representations. Dropout effectively trains multiple thinned networks, whose predictions are averaged during inference, leading to improved generalization. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout

F. **Enable early stopping:** Early stopping monitors the model's performance on a validation set during training. Training is stopped when the performance on the validation set starts to degrade (e.g., accuracy decreases or loss increases), even if the training accuracy continues to improve. This prevents the model from overfitting to the training data and ensures that the model with the best generalization performance is selected. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

Why the other options are incorrect:

A. **Choose a higher number of layers:** Increasing the number of layers would further exacerbate the overfitting problem. A more complex model would be even more prone to memorizing the training data.

C. **Choose a smaller learning rate:** While a smaller learning rate can help with convergence and prevent overshooting the optimal weights, it doesn't directly address overfitting. It might even increase overfitting if the model is trained for too long.

E. **Include all the images from the test set in the training set:** This is a data leakage issue. The model would be evaluated on data it has already seen, leading to an overly optimistic and unreliable assessment of its performance in real-world scenarios. It defeats the purpose of having a test set to evaluate the model's generalization ability.

Question: 95

Exam Heist

This graph shows the training and validation loss against the epochs for a neural network.
The network being trained is as follows:
- Two dense layers, one output neuron
- 100 neurons in each layer
- 100 epochs

Random initialization of weights

Which technique can be used to improve model performance in terms of accuracy in the validation set?

- Early stopping ✓
- Random initialization of weights with appropriate seed
- Increasing the number of epochs
- Adding another layer with the 100 neurons

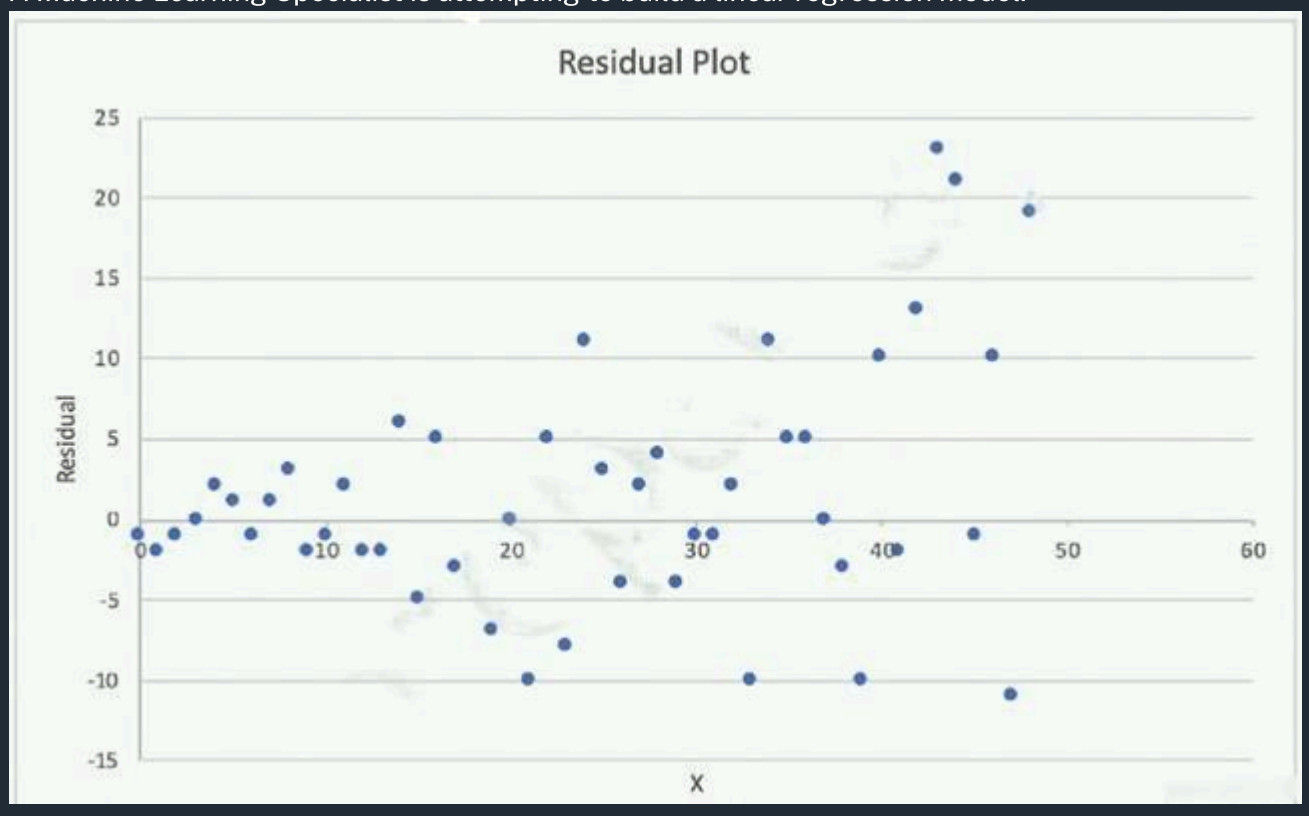
Explanation:

The answer is Early Stopping. Stopp the training before accuracy start do decrease.

Question: 96

Exam Heist

A Machine Learning Specialist is attempting to build a linear regression model.



Given the displayed residual plot only, what is the MOST likely problem with the model?

- Linear regression is inappropriate. The residuals do not have constant variance. ✓
- Linear regression is inappropriate. The underlying data has outliers.
- Linear regression is appropriate. The residuals have a zero mean.
- Linear regression is appropriate. The residuals have constant variance.

Explanation:
Linear regression is inappropriate. The residuals do not have constant variance.
Reference:
https://blog.minitab.com/en/the-statistics-game/checking-the-assumption-of-constant-variance-in-regression-analyses

Question: 97

Exam Heist

A large company has developed a BI application that generates reports and dashboards using data collected from various operational metrics. The company wants to provide executives with an enhanced experience so they can use natural language to get data from the reports. The company wants the executives to be able ask questions using written and spoken interfaces.
Which combination of services can be used to build this conversational interface? (Choose three.)

- [ ] Alexa for Business
- [ ] Amazon Connect
- [x] Amazon Lex ✓
- [ ] Amazon Polly
- [x] Amazon Comprehend ✓
- [x] Amazon Transcribe ✓

Explanation:
The correct answer is **CEF (Amazon Lex, Amazon Comprehend, Amazon Transcribe)**. Here's a detailed justification:

Amazon Lex: This service provides the core conversational interface, building chatbots that understand and respond to user input. It's necessary for handling both written and spoken queries, recognizing the user's intent and extracting relevant information. Lex uses Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) to interpret user input. https://aws.amazon.com/lex/
Amazon Comprehend: Comprehend is a natural language processing (NLP) service that is used to extract meaning and relationships from text. For this scenario, Comprehend will analyze the transcribed user questions to identify key entities, sentiments, and relationships to pass the query structured to the BI application. This helps in accurately interpreting the intent behind natural language queries. https://aws.amazon.com/comprehend/

Amazon Transcribe: Since the executives want to use spoken interfaces, Amazon Transcribe is crucial. It automatically converts speech to text, allowing the system to process spoken queries using Lex and Comprehend.
https://aws.amazon.com/transcribe/
Why the other options are not ideal:

Alexa for Business: Alexa for Business facilitates the integration of Alexa into the workplace. While it could be used to *access* the conversational interface, it doesn't provide the core NLP components needed for the application itself.
Amazon Connect: Amazon Connect is a cloud contact center service. It's relevant for customer service applications, but not directly applicable to building a natural language interface for BI reporting.
Amazon Polly: Amazon Polly converts text to speech. While useful for providing *spoken responses*, the prompt's core requirement focuses on understanding user input, not generating output. Polly is more beneficial for replying to users in a spoken form based on data retrieval from BI app.

Question: 98

Exam Heist

A machine learning specialist works for a fruit processing company and needs to build a system that categorizes apples into three types. The specialist has collected a dataset that contains 150 images for each type of apple and applied transfer learning on a neural network that was pretrained on ImageNet with this dataset.
The company requires at least 85% accuracy to make use of the model.
After an exhaustive grid search, the optimal hyperparameters produced the following:
↬ 68% accuracy on the training set
↬ 67% accuracy on the validation set
What can the machine learning specialist do to improve the system's accuracy?

- Upload the model to an Amazon SageMaker notebook instance and use the Amazon SageMaker HPO feature to optimize the model's hyperparameters.
- Add more data to the training set and retrain the model using transfer learning to reduce the bias. ✓
- Use a neural network model with more layers that are pretrained on ImageNet and apply transfer learning to increase the variance.
- Train a new model using the current neural network architecture.

Explanation:
Here's a detailed justification for why option B is the most appropriate solution to improve the apple classification system's accuracy, given the scenario.
The problem indicates a significant issue with the model's performance: both training and validation accuracy are low (around 67-68%). This suggests a high bias, also known as underfitting. Underfitting means the model isn't complex enough to capture the underlying patterns in the data. Grid search already attempted hyperparameter optimization, so further HPO (as in option A) might yield only marginal gains.
Option C suggests increasing the model's complexity (more layers) to increase variance. However, increasing model complexity *without* addressing the data issue can lead to overfitting, especially when the dataset is small.
Option D, retraining with the current architecture, won't solve the fundamental problem of insufficient data or features for the model to learn effectively.
Option B addresses the root cause of underfitting: insufficient data. Adding more data allows the model to learn more robust and generalizable patterns. Transfer learning helps leverage knowledge from ImageNet, but it still requires a sufficiently representative dataset for the specific apple classification task. A larger dataset would provide more examples of each apple type, enabling the model to better discriminate between them. Retraining after adding data would allow the model to adjust its weights and potentially achieve the desired 85% accuracy.
Therefore, the best approach is to add more data to the training set and retrain the model using transfer learning. This directly combats the underfitting issue.
Relevant Resources:

Transfer Learning: https://www.tensorflow.org/tutorials/images/transfer_learning (TensorFlow documentation explaining transfer learning concepts)
Bias and Variance: [https://www.analyticsvidhya.com/blog/2020/08/ Bias and Variance in Machine Learning](https://www.analyticsvidhya.com/blog/2020/08/Bias and Variance in Machine Learning) (Explanation of Bias-Variance Tradeoff)
Amazon SageMaker: https://aws.amazon.com/sagemaker/ (For general information about SageMaker if HPO is a secondary consideration after addressing data issue)

Question: 99

Exam Heist

A company uses camera images of the tops of items displayed on store shelves to determine which items were removed and which ones still remain. After several hours of data labeling, the company has a total of 1,000 hand-labeled images covering 10 distinct items. The training results were poor.
Which machine learning approach fulfills the company's long-term needs?

- Convert the images to grayscale and retrain the model
- Reduce the number of distinct items from 10 to 2, build the model, and iterate
- Attach different colored labels to each item, take the images again, and build the model
- Augment training data for each item using image variants like inversions and translations, build the model, and iterate. ✓

Explanation:
Here's a detailed justification for why option D is the most suitable approach for the company's machine learning challenge:
The primary issue highlighted is poor training results with only 1,000 labeled images across 10 distinct items. This indicates a significant lack of sufficient training data, leading to overfitting and poor generalization capabilities of the model. Simply converting images to grayscale (Option A) won't address the fundamental problem of insufficient data. Reducing the number of items (Option B), while potentially improving initial results, limits the long-term goal of identifying all 10 items. Using colored labels (Option C) could introduce biases and might not be robust to variations in lighting conditions or camera angles.
Data augmentation (Option D) directly tackles the data scarcity problem. By creating variations of existing images through techniques like rotations, inversions, translations, scaling, and color adjustments, the effective dataset size is significantly increased without requiring additional data labeling. This helps the model learn more robust and invariant features, improving its ability to generalize to unseen images. The iterative approach allows for continuous improvement and refinement of the model as more data or insights are gained.
Augmenting the training data helps reduce overfitting which is a common problem in machine learning when the model is trained on a limited dataset. By increasing the variety of the training data, the model is forced to learn more robust features and is less likely to memorize the training data. Image augmentation also addresses the problem of limited viewpoint and environmental variations in the training data. In real-world scenarios, images might be taken from different angles or under different lighting conditions. Data augmentation helps the model be more robust to these variations.
Furthermore, this approach aligns with best practices in computer vision, where data augmentation is a standard technique for improving model performance when training data is limited. The iterative build-and-improve aspect of this approach is a fundamental principle of machine learning model development, emphasizing continuous evaluation and refinement for optimal results.
Authoritative Links:

Data Augmentation: https://www.tensorflow.org/tutorials/images/data_augmentation
Overfitting: https://www.ibm.com/cloud/learn/overfitting

Question: 100 Exam Heist

A Data Scientist is developing a binary classifier to predict whether a patient has a particular disease on a series of test results. The Data Scientist has data on
400 patients randomly selected from the population. The disease is seen in 3% of the population.
Which cross-validation strategy should the Data Scientist adopt?

- A k-fold cross-validation strategy with k=5
- A stratified k-fold cross-validation strategy with k=5 ✓
- A k-fold cross-validation strategy with k=5 and 3 repeats
- An 80/20 stratified split between training and validation

Explanation:
The correct answer is **B. A stratified k-fold cross-validation strategy with k=5**. Here's why:
The problem describes a dataset with a class imbalance. Only 3% of the patients have the disease. In such scenarios, standard k-fold cross-validation can lead to folds where some might have very few or even zero positive cases (patients with the disease). This would result in a model trained without representative data for the minority class (patients *with* the disease), leading to poor generalization and biased performance metrics.
Stratified k-fold cross-validation addresses this issue. It ensures that each fold maintains the same class distribution as the original dataset. This means that each fold will have approximately 3% of patients with the disease, preventing any single fold from being completely devoid of positive cases. This guarantees a more representative evaluation of the model's performance, especially its ability to correctly identify the minority class (patients with the disease), which is crucial in this medical diagnosis scenario.

Option A, simple k-fold, is not suitable due to the potential for imbalanced folds. Option C, k-fold with repeats, doesn't inherently address the class imbalance problem; it just runs the flawed process multiple times. Option D, a single 80/20 stratified split, provides only one evaluation, potentially influenced by the specific data split. Cross-validation, specifically stratified k-fold, offers a more robust and reliable assessment of the model's generalization ability by averaging the performance across multiple folds, each maintaining the original class distribution. Using k=5 is a common practice and provides a good balance between computational cost and variance reduction in the performance estimate.
For further research, you can refer to these resources:

Scikit-learn documentation on StratifiedKFold: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html
Cross-validation techniques for imbalanced datasets: https://imbalanced-learn.org/stable/user_guide.html

Question: 101

Exam Heist

A technology startup is using complex deep neural networks and GPU compute to recommend the company's products to its existing customers based upon each customer's habits and interactions. The solution currently pulls each dataset from an Amazon S3 bucket before loading the data into a TensorFlow model pulled from the company's Git repository that runs locally. This job then runs for several hours while continually outputting its progress to the same S3 bucket. The job can be paused, restarted, and continued at any time in the event of a failure, and is run from a central queue.
Senior managers are concerned about the complexity of the solution's resource management and the costs involved in repeating the process regularly. They ask for the workload to be automated so it runs once a week, starting Monday and completing by the close of business Friday.
Which architecture should be used to scale the solution at the lowest cost?

- Implement the solution using AWS Deep Learning Containers and run the container as a job using AWS Batch on a GPU-compatible Spot Instance ✓
- Implement the solution using a low-cost GPU-compatible Amazon EC2 instance and use the AWS Instance Scheduler to schedule the task
- Implement the solution using AWS Deep Learning Containers, run the workload using AWS Fargate running on Spot Instances, and then schedule the task using the built-in task scheduler
- Implement the solution using Amazon ECS running on Spot Instances and schedule the task using the ECS service scheduler

Explanation:
Here's a detailed justification for why option A is the best solution, considering the cost-effectiveness and requirements of the given scenario:
The key requirements are automation, weekly execution with a deadline, fault tolerance (pause/restart), and cost optimization for a GPU-intensive deep learning workload.
Option A leverages several AWS services optimally. AWS Deep Learning Containers (DLCs) provide pre-configured environments optimized for deep learning frameworks like TensorFlow, eliminating the need for manual setup and ensuring consistency. AWS Batch is designed for batch computing workloads and can automatically provision and manage the necessary compute resources. Running the Batch job on GPU-compatible Spot Instances significantly reduces costs compared to on-demand instances, as Spot Instances offer spare EC2 compute capacity at discounted rates. Batch can also handle retries and dependencies, catering to the fault tolerance requirement. Batch natively integrates with S3 and can pull data directly, streamlining the process.
Option B, using a single EC2 instance and Instance Scheduler, doesn't effectively address scalability. While cost-effective for constant usage, it's not ideal for intermittent, high-demand GPU workloads. Instance Scheduler simply starts and stops the instance, not managing the actual job execution or providing inherent fault tolerance besides starting the process again. It lacks the orchestration capabilities of Batch.
Option C, using AWS Fargate, is typically more expensive than EC2 Spot Instances for long-running, compute-intensive workloads like this one. Fargate abstracts away the underlying infrastructure, which leads to higher operational cost with less ability to save money by purchasing and taking advantage of compute savings plans. Also, Fargate Spot is less mature and offers fewer instance options compared to EC2 Spot. Fargate is not typically used with spot instances, though it is possible. The built-in task scheduler is less robust than AWS Batch for complex workflows.
Option D, using ECS with Spot Instances and ECS service scheduler, also introduces complexity. ECS service scheduler is best suited for maintaining a desired state of running containers, not for orchestrating batch jobs with dependencies and retries. While ECS can run on Spot Instances, Batch offers better integration for batch processing and manages the Spot Instance lifecycle more effectively in this context.
AWS Batch efficiently manages resources, including GPU instances, optimizing costs while ensuring the job completes within the deadline, and is therefore the most appropriate selection.
Relevant Links:

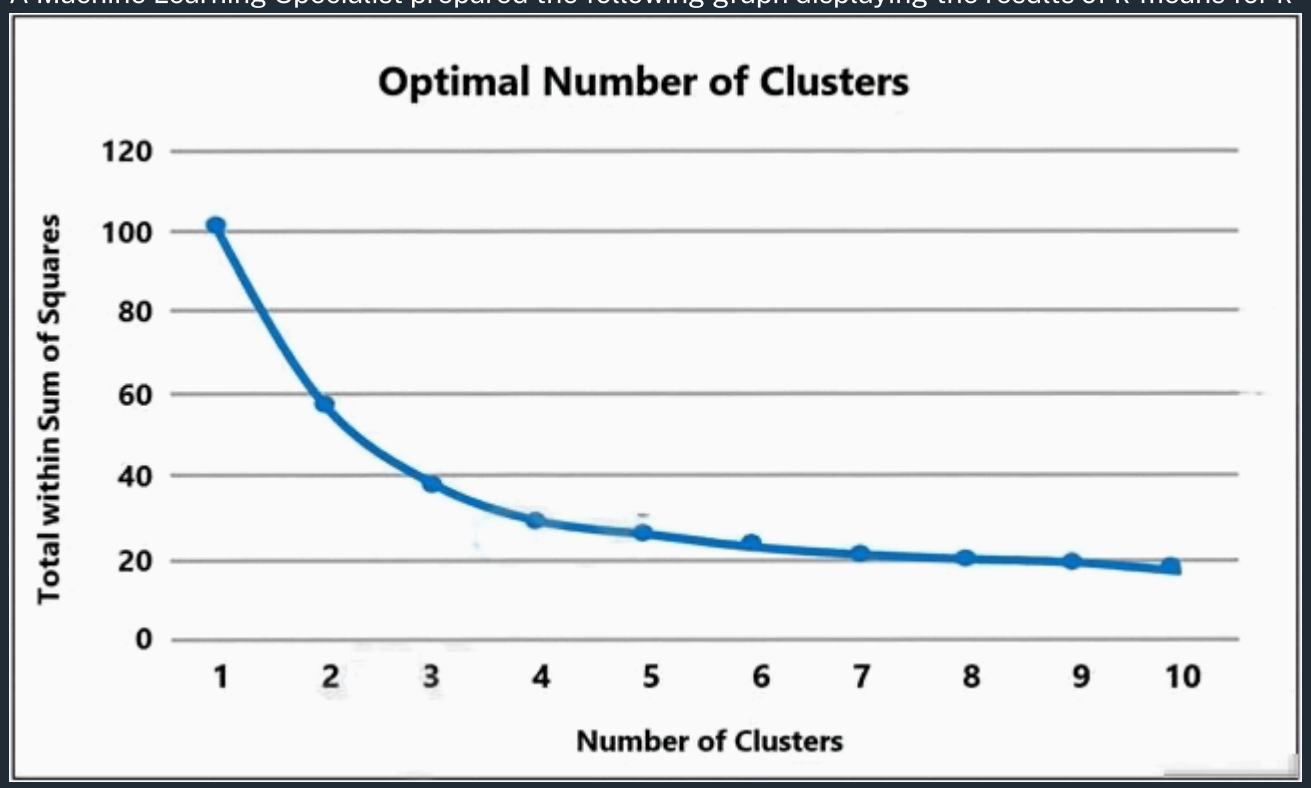
AWS Batch: https://aws.amazon.com/batch/
AWS Deep Learning Containers: https://aws.amazon.com/machine-learning/containers/

Amazon EC2 Spot Instances: https://aws.amazon.com/ec2/spot/

Question: 102

Exam Heist

A Machine Learning Specialist prepared the following graph displaying the results of k-means for k = [1..10]:



Considering the graph, what is a reasonable selection for the optimal choice of k?

- 1
- 4 ✓
- 7
- 10

Explanation:
Elbow is more visible at 4.
Reference:
https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/

Question: 103

Exam Heist

A media company with a very large archive of unlabeled images, text, audio, and video footage wishes to index its assets to allow rapid identification of relevant content by the Research team. The company wants to use machine learning to accelerate the efforts of its in-house researchers who have limited machine learning expertise.
Which is the FASTEST route to index the assets?

- Use Amazon Rekognition, Amazon Comprehend, and Amazon Transcribe to tag data into distinct categories/classes. ✓
- Create a set of Amazon Mechanical Turk Human Intelligence Tasks to label all footage.
- Use Amazon Transcribe to convert speech to text. Use the Amazon SageMaker Neural Topic Model (NTM) and Object Detection algorithms to tag data into distinct categories/classes.
- Use the AWS Deep Learning AMI and Amazon EC2 GPU instances to create custom models for audio transcription and topic modeling, and use object detection to tag data into distinct categories/classes.

Explanation:
The fastest route to index the media company's assets, given limited machine learning expertise and a need for rapid results, is option A: using Amazon Rekognition, Amazon Comprehend, and Amazon Transcribe. These are managed AI services that provide pre-trained models for image and video analysis (Rekognition), natural language processing (Comprehend), and speech-to-text conversion (Transcribe).
Option B, using Amazon Mechanical Turk, involves human labeling, which is a time-consuming and expensive process, contrasting with the requirement for speed. Option C, while utilizing Transcribe for speech-to-text, unnecessarily introduces SageMaker's Neural Topic Model (NTM) and Object Detection algorithms, increasing complexity without a clear need. While

SageMaker NTM can be useful, applying pre-trained models for general topic extraction using Comprehend is faster and more straightforward. Object detection, if needed, is provided within Rekognition. Furthermore, managing SageMaker jobs necessitates machine learning expertise, which the researchers lack.
Option D necessitates creating custom models on EC2 instances using a Deep Learning AMI. This approach demands substantial machine learning proficiency, significant development time, and computational resources, directly contradicting the need for a fast solution and the researchers' lack of expertise. Pre-trained models using Rekognition, Comprehend, and Transcribe significantly reduce the required effort and time to implement. These services offer a readily available and easy-to-use API, empowering rapid tagging and indexing for immediate use by the Research team. These services abstract away the underlying complexities of the machine learning models, allowing the company to focus on utilizing the insights derived from the data.

Amazon Rekognition: https://aws.amazon.com/rekognition/
Amazon Comprehend: https://aws.amazon.com/comprehend/
Amazon Transcribe: https://aws.amazon.com/transcribe/

Question: 104

Exam Heist

A Machine Learning Specialist is working for an online retailer that wants to run analytics on every customer visit, processed through a machine learning pipeline.
The data needs to be ingested by Amazon Kinesis Data Streams at up to 100 transactions per second, and the JSON data blob is 100 KB in size.
What is the MINIMUM number of shards in Kinesis Data Streams the Specialist should use to successfully ingest this data?

- 1 shards
- 10 shards ✓
- 100 shards
- 1,000 shards

Explanation:
Here's a detailed justification for why the answer is B (10 shards) for the Kinesis Data Streams sizing question:
Kinesis Data Streams capacity is determined by the number of shards provisioned. Each shard can support 1 MB/second of data input and 1,000 records/second of data input. In this scenario, the retailer needs to ingest 100 transactions per second, where each transaction (JSON blob) is 100 KB in size.
First, let's calculate the total data ingestion rate in MB/second: 100 transactions/second * 100 KB/transaction = 10,000 KB/second. Convert this to MB/second: 10,000 KB/second / 1024 KB/MB ≈ 9.77 MB/second.
Each shard supports 1 MB/second. Therefore, to handle 9.77 MB/second, the system requires at least 9.77 MB/second / 1 MB/second/shard = 9.77 shards. Since you cannot provision fractional shards, the number must be rounded up to the nearest whole number, which is 10 shards.
The record limit is also a factor, but in this case is not the constraint. 100 records per second is less than the 1000 records/second per shard limit. Thus the data throughput is the limiting factor.
Therefore, the minimum number of shards required to successfully ingest the data is 10. Using fewer shards would cause throttling and data loss.
Authoritative Links:

Amazon Kinesis Data Streams Limits: https://docs.aws.amazon.com/kinesis/latest/dev/service-sizes-limits.html
Scaling Kinesis Data Streams: https://aws.amazon.com/blogs/big-data/scaling-amazon-kinesis-data-streams-for-optimal-cost-efficiency/

Question: 105

Exam Heist

A Machine Learning Specialist is deciding between building a naive Bayesian model or a full Bayesian network for a classification problem. The Specialist computes the Pearson correlation coefficients between each feature and finds that their absolute values range between 0.1 to 0.95.
Which model describes the underlying data in this situation?

- A naive Bayesian model, since the features are all conditionally independent.
- A full Bayesian network, since the features are all conditionally independent.
- A naive Bayesian model, since some of the features are statistically dependent.
- A full Bayesian network, since some of the features are statistically dependent. ✓

Explanation:

The correct answer is D: A full Bayesian network, since some of the features are statistically dependent. Here's why:

Naive Bayes classifiers operate under the strong assumption of feature independence. This means they assume that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. In simpler terms, the features are conditionally independent given the class.

The problem description states that the Pearson correlation coefficients between the features range from 0.1 to 0.95 (absolute values). A Pearson correlation coefficient measures the linear relationship between two variables. A value close to 0 indicates a weak or no linear relationship, while values closer to 1 or -1 indicate a strong linear relationship. The given range clearly indicates that some features exhibit a substantial correlation, and therefore statistical dependence, with each other.

Since the features are not independent, the key assumption of the naive Bayes classifier is violated. Applying a naive Bayes model when this assumption is violated can lead to suboptimal performance and inaccurate predictions.

A full Bayesian network, on the other hand, does not assume feature independence. It explicitly models the dependencies between variables using a directed acyclic graph (DAG). This allows it to capture complex relationships and dependencies present in the data, including the correlations observed in this case. The network structure represents conditional dependencies between variables. Therefore, it is more appropriate to use a full Bayesian network because it can accommodate the observed feature dependencies, leading to potentially more accurate and reliable results.

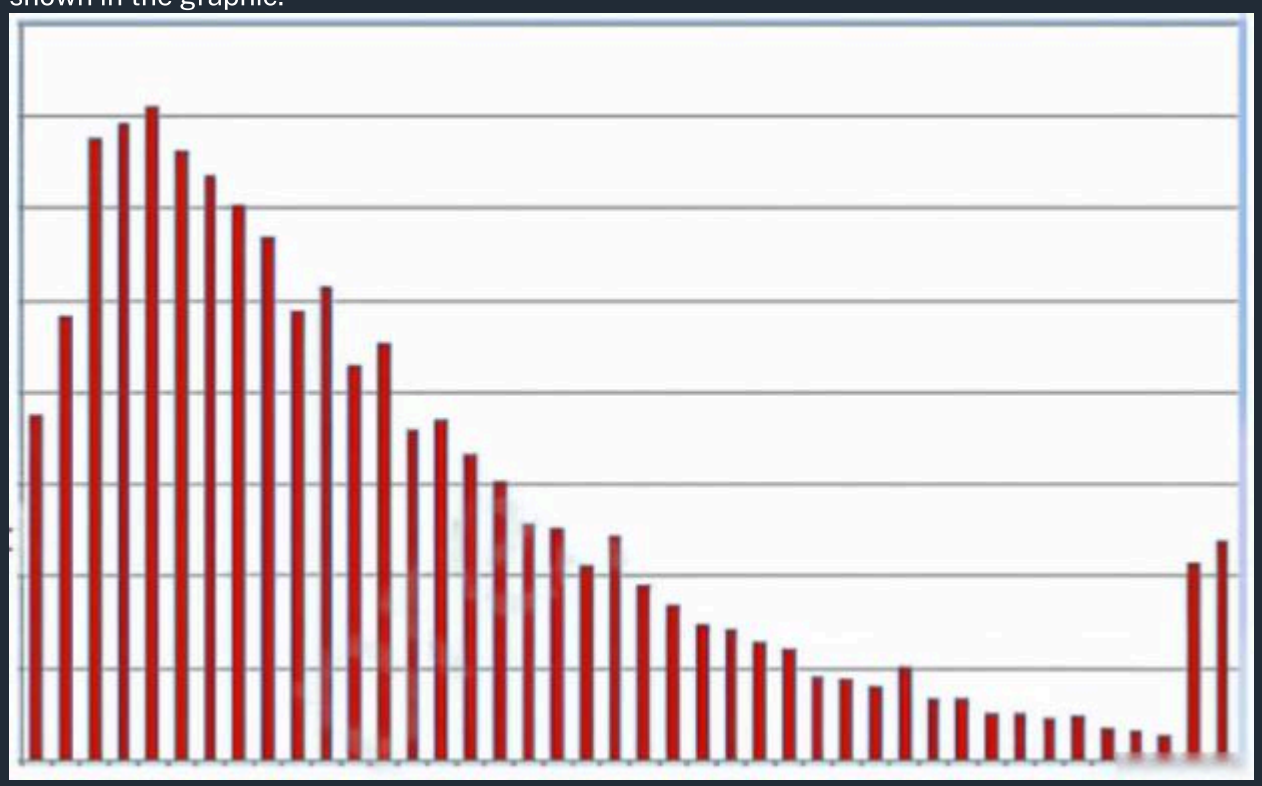
Further research:

Bayesian Networks: https://www.cs.princeton.edu/courses/archive/fall10/cos424/scribe_notes/1122.pdf
Naive Bayes Classifier: https://scikit-learn.org/stable/modules/naive_bayes.html

Question: 106

Exam Heist

A Data Scientist is building a linear regression model and will use resulting p-values to evaluate the statistical significance of each coefficient. Upon inspection of the dataset, the Data Scientist discovers that most of the features are normally distributed. The plot of one feature in the dataset is shown in the graphic.



What transformation should the Data Scientist apply to satisfy the statistical assumptions of the linear regression model?

- Exponential transformation
- Logarithmic transformation ✓
- Polynomial transformation
- Sinusoidal transformation

Explanation:

The linear regression model assumes that the errors are normally distributed. The plot of the feature shows that the errors are not normally distributed.

The logarithmic transformation can be used to transform the errors to be normally distributed.

The exponential transformation, polynomial transformation, and sinusoidal transformation cannot be used to transform the errors to be normally distributed.

Question: 107

Exam Heist

A Machine Learning Specialist is assigned to a Fraud Detection team and must tune an XGBoost model, which is working appropriately for test data. However, with unknown data, it is not working as expected. The existing parameters are provided as follows.

```
param = {
      'eta': 0.05,  # the training step for each iteration
      'silent': 1,  # logging mode - quiet
      'n_estimators':2000,
      'max_depth':30,
      'min_child_weight': 3,
      'gamma': 0,
      'subsample': 0.8,
      'objective': 'multi:softprob',  # error evaluation for multiclass training
      'num_class': 201}  # the number of classes that exist in this dataset
num_round = 60  # the number of training iterations
```

Which parameter tuning guidelines should the Specialist follow to avoid overfitting?

- Increase the max_depth parameter value.
- Lower the max_depth parameter value. ✓
- Update the objective to binary:logistic.
- Lower the min_child_weight parameter value.

Explanation:

B lower max_depth is the correct answer. D min_child_weight means something like "stop trying to split once your sample size in a node goes below a given threshold"Lower min_child_weight, the tree becomes more deep and complex.Increase min_child_weight, the tree will have less branches and less complexity.

Question: 108

Exam Heist

A data scientist is developing a pipeline to ingest streaming web traffic data. The data scientist needs to implement a process to identify unusual web traffic patterns as part of the pipeline. The patterns will be used downstream for alerting and incident response. The data scientist has access to unlabeled historic data to use, if needed.
The solution needs to do the following:
Calculate an anomaly score for each web traffic entry.
Adapt unusual event identification to changing web patterns over time.

Which approach should the data scientist implement to meet these requirements?

- Use historic web traffic data to train an anomaly detection model using the Amazon SageMaker Random Cut Forest (RCF) built-in model. Use an Amazon Kinesis Data Stream to process the incoming web traffic data. Attach a preprocessing AWS Lambda function to perform data enrichment by calling the RCF model to calculate the anomaly score for each record.
- Use historic web traffic data to train an anomaly detection model using the Amazon SageMaker built-in XGBoost model. Use an Amazon Kinesis Data Stream to process the incoming web traffic data. Attach a preprocessing AWS Lambda function to perform data enrichment by calling the XGBoost model to calculate the anomaly score for each record.
- Collect the streaming data using Amazon Kinesis Data Firehose. Map the delivery stream as an input source for Amazon Kinesis Data Analytics. Write a SQL query to run in real time against the streaming data with the k-Nearest Neighbors (kNN) SQL extension to calculate anomaly scores for each record using a tumbling window.
- Collect the streaming data using Amazon Kinesis Data Firehose. Map the delivery stream as an input source for Amazon Kinesis Data Analytics. Write a SQL query to run in real time against the streaming data with the Amazon Random Cut Forest (RCF) SQL extension to calculate anomaly scores for each record using a sliding window. ✓

Explanation:

Collect the streaming data using Amazon Kinesis Data Firehose. Map the delivery stream as an input source for Amazon Kinesis Data Analytics. Write a SQL query to run in real time against the streaming data with the Amazon Random Cut Forest (RCF) SQL extension to calculate anomaly scores for each record using a sliding window.

Question: 109

Exam Heist

A Data Scientist received a set of insurance records, each consisting of a record ID, the final outcome among 200 categories, and the date of the final outcome.
Some partial information on claim contents is also provided, but only for a few of the 200 categories. For each outcome category, there are hundreds of records distributed over the past 3 years. The Data Scientist wants to predict how many claims to expect in each category from month to month, a few months in advance.
What type of machine learning model should be used?

- Classification month-to-month using supervised learning of the 200 categories based on claim contents.
- Reinforcement learning using claim IDs and timestamps where the agent will identify how many claims in each category to expect from month to month.
- Forecasting using claim IDs and timestamps to identify how many claims in each category to expect from month to month. ✓
- Classification with supervised learning of the categories for which partial information on claim contents is provided, and forecasting using claim IDs and timestamps for all other categories.

Explanation:
The Data Scientist aims to predict the number of claims in each of 200 categories on a monthly basis, a few months into the future. This problem is fundamentally about predicting future values based on historical time-series data. Therefore, forecasting techniques are the most appropriate.
Option C, forecasting using claim IDs and timestamps, correctly identifies this. It utilizes the temporal information (timestamps) associated with the claims to understand trends and seasonality in claim frequency within each category. This approach directly addresses the objective of predicting the *number* of claims over time. Claim IDs are likely used to simply count the number of claims over that period.
Option A (classification) is inappropriate because it focuses on categorizing claims based on content. While potentially useful, it doesn't directly predict *how many* claims to expect each month.
Option B (reinforcement learning) is overkill and doesn't naturally fit the problem structure. Reinforcement learning is best suited for situations where an agent learns through trial and error, receiving rewards or penalties for its actions. This doesn't align with the goal of predicting claim volumes.
Option D is a hybrid approach that combines classification and forecasting. While potentially valid, it's unnecessarily complex. The problem can be solved more efficiently and directly using forecasting techniques for *all* categories, leveraging historical time-series data. The claim content information, if relevant, could be incorporated as features in a forecasting model.
Therefore, forecasting (Option C) provides the most direct and efficient way to address the problem of predicting the number of claims in each category from month to month. Time series forecasting models are designed for this type of prediction based on past trends.
Further Research:

Time Series Forecasting: https://otexts.com/fpp2/ (Online textbook on forecasting)
Amazon Forecast: https://aws.amazon.com/forecast/ (AWS managed service for time-series forecasting)
Time Series Analysis with AWS: https://aws.amazon.com/blogs/big-data/time-series-analysis-on-aws-part-1-data-ingestion-and-storage/ (AWS blog about time-series analysis)

Question: 110

Exam Heist

A company that promotes healthy sleep patterns by providing cloud-connected devices currently hosts a sleep tracking application on AWS. The application collects device usage information from device users. The company's Data Science team is building a machine learning model to predict if and when a user will stop utilizing the company's devices. Predictions from this model are used by a downstream application that determines the best approach for contacting users.
The Data Science team is building multiple versions of the machine learning model to evaluate each version against the company's business goals. To measure long-term effectiveness, the team wants to run multiple versions of the model in parallel for long periods of time, with the ability to control the portion of inferences served by the models.
Which solution satisfies these requirements with MINIMAL effort?

- Build and host multiple models in Amazon SageMaker. Create multiple Amazon SageMaker endpoints, one for each model. Programmatically control invoking different models for inference at the application layer.
- Build and host multiple models in Amazon SageMaker. Create an Amazon SageMaker endpoint configuration with multiple production variants. Programmatically control the portion of the inferences served by the multiple models by updating the endpoint configuration. ✓
- Build and host multiple models in Amazon SageMaker Neo to take into account different types of medical devices. Programmatically control which model is invoked for inference based on the medical device type.
- Build and host multiple models in Amazon SageMaker. Create a single endpoint that accesses multiple models. Use Amazon SageMaker batch transform to control invoking the different models through the single endpoint.

Explanation:
The correct answer is **B**. Here's a detailed justification:

The scenario requires A/B testing of multiple model versions over a long period, with the ability to control the traffic distribution. Amazon SageMaker's endpoint configurations with multiple production variants are ideally suited for this purpose. This feature directly supports traffic splitting and management between different model versions deployed behind a single endpoint. By adjusting the weights assigned to each production variant in the endpoint configuration, you can precisely control the percentage of inference requests each model serves.
Option A involves managing multiple endpoints and implementing the traffic routing logic in the application layer. This adds significant complexity to the application and necessitates extra code for load balancing and monitoring.
Option C suggests using SageMaker Neo, which focuses on optimizing models for different hardware platforms. While optimization is useful, it does not directly address the requirement of A/B testing and traffic splitting. The "different types of medical devices" mention is a distraction, as the core requirement is long-term A/B testing.
Option D proposes using SageMaker Batch Transform. Batch Transform is designed for offline inference on large datasets, not real-time, low-latency serving required by the application. The goal is to provide an API for a downstream application.
SageMaker's production variants provide a built-in mechanism for A/B testing and model monitoring without the complexity of managing multiple endpoints or custom routing logic. This approach aligns with the "MINIMAL effort" requirement stated in the question. Updating the endpoint configuration is a relatively simple operation, allowing the Data Science team to dynamically adjust traffic allocation as needed.
For further research, refer to the official AWS documentation:

Deploy a Model on Amazon SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
Amazon SageMaker Endpoints: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-endpoints.html
CreateEndpointConfig: https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_CreateEndpointConfig.html

Question: 111

Exam Heist

An agricultural company is interested in using machine learning to detect specific types of weeds in a 100-acre grassland field. Currently, the company uses tractor-mounted cameras to capture multiple images of the field as 10 — 10 grids. The company also has a large training dataset that consists of annotated images of popular weed classes like broadleaf and non-broadleaf docks.
The company wants to build a weed detection model that will detect specific types of weeds and the location of each type within the field. Once the model is ready, it will be hosted on Amazon SageMaker endpoints. The model will perform real-time inferencing using the images captured by the cameras.
Which approach should a Machine Learning Specialist take to obtain accurate predictions?

- ○ Prepare the images in RecordIO format and upload them to Amazon S3. Use Amazon SageMaker to train, test, and validate the model using an image classification algorithm to categorize images into various weed classes.
- ○ Prepare the images in Apache Parquet format and upload them to Amazon S3. Use Amazon SageMaker to train, test, and validate the model using an object-detection single-shot multibox detector (SSD) algorithm.
- ○ Prepare the images in RecordIO format and upload them to Amazon S3. Use Amazon SageMaker to train, test, and validate the model using an object-detection single-shot multibox detector (SSD) algorithm. ✓
- ○ Prepare the images in Apache Parquet format and upload them to Amazon S3. Use Amazon SageMaker to train, test, and validate the model using an image classification algorithm to categorize images into various weed classes.

Explanation:
Here's a detailed justification for why option C is the best approach for the agricultural company's weed detection problem:
The problem requires identifying specific types of weeds *and* their location within the field. This necessitates an object detection approach, not just image classification. Image classification, as suggested in options A and D, would only categorize the *entire* image as belonging to a specific weed class, without pinpointing the exact location of each weed within the grid.
Object detection, specifically using algorithms like Single Shot Multibox Detector (SSD), as mentioned in options B and C, is designed to identify multiple objects within an image and provide bounding boxes around them, fulfilling the requirement of detecting both the type and location. SSD is particularly suitable for real-time inferencing due to its speed.
Now, concerning the data format, RecordIO is a binary format optimized for image data and is natively supported by Amazon SageMaker for training image-based models. Apache Parquet, while a columnar storage format ideal for analytical queries, isn't typically used for directly feeding image data into training algorithms. Therefore, RecordIO is the preferred format for this use case.
Consequently, option C — preparing images in RecordIO format, storing them in S3, and utilizing SageMaker with an object-detection SSD algorithm — provides the most accurate and efficient solution. It addresses both the need for object localization and utilizes a data format optimized for SageMaker's image-based training.
Relevant Links:

Amazon SageMaker Object Detection: https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection.html
RecordIO format: While AWS documentation doesn't explicitly say SageMaker *requires* RecordIO, it's commonly used and efficiently handled.

SSD Algorithm: https://paperswithcode.com/method/ssd (For the algorithm specifics). Though not an AWS link, it describes the algorithm mentioned in the possible answers.

Question: 112

Exam Heist

A manufacturer is operating a large number of factories with a complex supply chain relationship where unexpected downtime of a machine can cause production to stop at several factories. A data scientist wants to analyze sensor data from the factories to identify equipment in need of preemptive maintenance and then dispatch a service team to prevent unplanned downtime. The sensor readings from a single machine can include up to 200 data points including temperatures, voltages, vibrations, RPMs, and pressure readings.
To collect this sensor data, the manufacturer deployed Wi-Fi and LANs across the factories. Even though many factory locations do not have reliable or high-speed internet connectivity, the manufacturer would like to maintain near-real-time inference capabilities.
Which deployment architecture for the model will address these business requirements?

- Deploy the model in Amazon SageMaker. Run sensor data through this model to predict which machines need maintenance.
- Deploy the model on AWS IoT Greengrass in each factory. Run sensor data through this model to infer which machines need maintenance. ✓
- Deploy the model to an Amazon SageMaker batch transformation job. Generate inferences in a daily batch report to identify machines that need maintenance.
- Deploy the model in Amazon SageMaker and use an IoT rule to write data to an Amazon DynamoDB table. Consume a DynamoDB stream from the table with an AWS Lambda function to invoke the endpoint.

Explanation:
Here's a detailed justification for why option B is the best solution for the manufacturer's needs, along with links for further research:
The core requirement is to perform near-real-time inference for predictive maintenance, even with unreliable internet connectivity at some factory locations. This points directly to edge computing solutions.
Option B: Deploy the model on AWS IoT Greengrass in each factory. This is the most suitable approach because:

Edge Computing: AWS IoT Greengrass allows you to deploy machine learning models directly onto devices within the factory (the "edge"). This enables local inference without relying on a constant internet connection. This addresses the scenario where some factories lack reliable high-speed internet.
Near-Real-Time Inference: Inference happens locally and quickly. This satisfies the near-real-time requirement for identifying equipment needing preemptive maintenance and minimizing downtime.
Reduced Latency and Bandwidth: Processing data locally minimizes latency compared to sending all data to the cloud for inference. Also, sending all sensor data to the cloud would consume large amounts of bandwidth and can result in network congestion.
Resiliency: Even if the internet connection is temporarily lost, the local Greengrass deployment continues to operate, ensuring continued monitoring and alerting.
Security: IoT Greengrass provides security mechanisms for your devices and data both at rest and in transit.
Why other options are less suitable:

Option A: Deploy the model in Amazon SageMaker. While SageMaker is excellent for model training and deployment, relying solely on SageMaker for inference requires a stable internet connection from all factories. This contradicts the unreliable connectivity constraint.
Option C: Deploy the model to an Amazon SageMaker batch transformation job. Batch transformation is not near-real-time. Generating daily reports is too infrequent for preemptive maintenance; downtime events could occur before the report is generated.
Option D: Deploy the model in Amazon SageMaker and use an IoT rule to write data to an Amazon DynamoDB table. Consume a DynamoDB stream from the table with an AWS Lambda function to invoke the endpoint. Like Option A, this depends on continuous internet connectivity to send data to DynamoDB and invoke the SageMaker endpoint via Lambda. The DynamoDB stream adds unnecessary complexity and potential latency compared to local inference. Also, this still requires reliable network connection.
In summary, AWS IoT Greengrass provides the required edge computing capabilities to meet the manufacturer's needs for near-real-time inference, even with intermittent internet connectivity.
Authoritative Links for Further Research:

AWS IoT Greengrass: https://aws.amazon.com/greengrass/
Edge Computing on AWS: https://aws.amazon.com/what-is/edge-computing/
Predictive Maintenance with AWS IoT: https://aws.amazon.com/solutions/implementations/predictive-maintenance/

Question: 113

Exam Heist

A Machine Learning Specialist is designing a scalable data storage solution for Amazon SageMaker. There is an existing TensorFlow-based model implemented as a train.py script that relies on static training data that is currently stored as TFRecords.
Which method of providing training data to Amazon SageMaker would meet the business requirements with the LEAST development overhead?

- Use Amazon SageMaker script mode and use train.py unchanged. Point the Amazon SageMaker training invocation to the local path of the data without reformatting the training data.
- Use Amazon SageMaker script mode and use train.py unchanged. Put the TFRecord data into an Amazon S3 bucket. Point the Amazon SageMaker training invocation to the S3 bucket without reformatting the training data. ✓
- Rewrite the train.py script to add a section that converts TFRecords to protobuf and ingests the protobuf data instead of TFRecords.
- Prepare the data in the format accepted by Amazon SageMaker. Use AWS Glue or AWS Lambda to reformat and store the data in an Amazon S3 bucket.

Explanation:
Option B is the most efficient and practical solution because it leverages the existing infrastructure and requires minimal code changes. SageMaker's script mode allows you to run your existing `train.py` script without modification. Since the script already handles TFRecords, there's no need to rewrite it or convert the data. S3 is a scalable and cost-effective storage solution that integrates seamlessly with SageMaker. By pointing the training job directly to the S3 bucket containing the TFRecords, SageMaker can access and process the data without requiring additional data transformation steps. This avoids the overhead of rewriting the script (Option C) or reformatting the data using Glue/Lambda (Option D). Using a local path (Option A) would negate the scalability benefits of SageMaker, as the data wouldn't be accessible to distributed training instances. Therefore, option B provides the best balance of minimal development overhead and scalability, aligning with the business requirements.
Further Research:

Amazon SageMaker Script Mode: https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo.html
Amazon S3: https://aws.amazon.com/s3/

Question: 114

Exam Heist

The chief editor for a product catalog wants the research and development team to build a machine learning system that can be used to detect whether or not individuals in a collection of images are wearing the company's retail brand. The team has a set of training data.
Which machine learning algorithm should the researchers use that BEST meets their requirements?

- Latent Dirichlet Allocation (LDA)
- Recurrent neural network (RNN)
- K-means
- Convolutional neural network (CNN) ✓

Explanation:
The correct answer is D, Convolutional Neural Network (CNN). Here's why:
The task is to identify if individuals in images are wearing a specific brand. This is an image classification problem. CNNs are specifically designed for image analysis and computer vision tasks. They excel at automatically learning hierarchical features from images, such as edges, textures, and patterns, which are essential for identifying objects (in this case, clothing with the brand's logo or design).
Let's examine the other options:

A. Latent Dirichlet Allocation (LDA): LDA is a topic modeling technique used for discovering abstract "topics" in a collection of documents. It's not suitable for image analysis.
B. Recurrent neural network (RNN): RNNs are primarily designed for sequential data, like text or time series. While they can be adapted for image analysis, they are not as effective as CNNs for general image classification tasks.
C. K-means: K-means is a clustering algorithm used for grouping data points based on similarity. It's an unsupervised learning technique and would not be appropriate for classifying images based on a specific brand (a supervised task where the model learns from labeled data).
In summary, CNN's ability to automatically extract relevant features from images makes them ideal for this branding detection use case, offering the highest accuracy and efficiency.For deeper understanding on CNN, you can refer these resources:

Stanford CS231n: Convolutional Neural Networks for Visual Recognition: https://cs231n.github.io/convolutional-networks/

TensorFlow documentation on CNNs: https://www.tensorflow.org/tutorials/images/cnn

Question: 115

Exam Heist

A retail company is using Amazon Personalize to provide personalized product recommendations for its customers during a marketing campaign. The company sees a significant increase in sales of recommended items to existing customers immediately after deploying a new solution version, but these sales decrease a short time after deployment. Only historical data from before the marketing campaign is available for training.
How should a data scientist adjust the solution?

- ◯ Use the event tracker in Amazon Personalize to include real-time user interactions. ✓
- ◯ Add user metadata and use the HRNN-Metadata recipe in Amazon Personalize.
- ◯ Implement a new solution using the built-in factorization machines (FM) algorithm in Amazon SageMaker.
- ◯ Add event type and event value fields to the interactions dataset in Amazon Personalize.

Explanation:

The correct answer is **A. Use the event tracker in Amazon Personalize to include real-time user interactions.**
Here's a detailed justification:
The problem describes a "cold start" scenario followed by a decay in performance. Initially, the model performs well because recommendations are based on historical data, which might reflect general preferences. However, as users interact with the marketing campaign and the recommendations, their preferences evolve. The model, trained solely on historical data *before* the campaign, fails to adapt to these real-time changes.
Amazon Personalize's event tracker is specifically designed to capture real-time user interactions like clicks, purchases, and adds-to-cart. By integrating the event tracker, the model can continuously learn from these real-time events. This allows the model to adapt to shifting user preferences *during* the campaign. This addresses the issue of decaying sales as user preferences diverge from the initial historical data used to train the model.
Option B, adding user metadata and using HRNN-Metadata, might improve initial recommendation quality, but it won't address the dynamic change in user preferences during the campaign. HRNN-Metadata primarily leverages historical user and item attributes, not real-time behavior.
Option C, switching to a factorization machine in SageMaker, is a more complex and less directly relevant solution. While FM can be useful in certain scenarios, Personalize is designed specifically for recommendation tasks and handles many implementation details for you. The core issue is the lack of real-time data, not necessarily the underlying algorithm. Re-implementing using SageMaker does not directly solve the root problem of not incorporating real-time interaction data.
Option D, adding event type and value fields to the interactions dataset, is partially correct in that event tracking can provide such data. However, it doesn't fully resolve the problem of not using *real-time* event data for continuous learning. The event tracker automatically manages the ingestion and utilization of these event types in real time, which is more efficient than manually updating the historical interactions dataset.
By leveraging the event tracker, Amazon Personalize can move beyond static historical data and become a dynamic, adaptive recommendation engine that keeps up with evolving user preferences during the campaign.
Refer to these official AWS resources for more information:

Amazon Personalize Event Tracker: https://docs.aws.amazon.com/personalize/latest/dg/recording-events.html
Amazon Personalize Recipes: https://docs.aws.amazon.com/personalize/latest/dg/working-with-predefined-recipes.html

Question: 116

Exam Heist

A machine learning (ML) specialist wants to secure calls to the Amazon SageMaker Service API. The specialist has configured Amazon VPC with a VPC interface endpoint for the Amazon SageMaker Service API and is attempting to secure traffic from specific sets of instances and IAM users. The VPC is configured with a single public subnet.
Which combination of steps should the ML specialist take to secure the traffic? (Choose two.)

- ☐ Add a VPC endpoint policy to allow access to the IAM users. ✓
- ☐ Modify the users' IAM policy to allow access to Amazon SageMaker Service API calls only.
- ☐ Modify the security group on the endpoint network interface to restrict access to the instances. ✓
- ☐ Modify the ACL on the endpoint network interface to restrict access to the instances.
- ☐ Add a SageMaker Runtime VPC endpoint interface to the VPC.

Explanation:

Here's a detailed justification for why options A and C are the correct choices for securing traffic to the Amazon SageMaker Service API using a VPC endpoint, and why the other options are incorrect:

A. Add a VPC endpoint policy to allow access to the IAM users.

A VPC endpoint policy is crucial for securing access through the endpoint. By default, a VPC endpoint grants full access to the service it connects to. A policy allows you to refine this and grant access only to specific IAM users, roles, or resources based on your security requirements. This principle of least privilege ensures that only authorized entities can use the SageMaker Service API through the VPC endpoint. Without a restrictive policy, any entity within the VPC with sufficient IAM permissions could use the endpoint, which defeats the purpose of targeted access control.

B. Modify the users' IAM policy to allow access to Amazon SageMaker Service API calls only.

While modifying IAM policies is important for overall security, it's not *specifically* relevant to securing traffic *through the VPC endpoint*. IAM policies control what a user can do regardless of where they are accessing the service from. While limiting a user to only SageMaker API calls is a good general security practice, it doesn't tie access to the VPC endpoint itself. The problem statement asks how to secure the traffic specifically *from* the specific set of instances and users utilizing the VPC endpoint.

C. Modify the security group on the endpoint network interface to restrict access to the instances.

Security groups act as virtual firewalls for network interfaces within your VPC. By modifying the security group associated with the endpoint's network interface, you can control which instances (specifically, which network interfaces belonging to those instances) are allowed to send traffic to the SageMaker Service API through that endpoint. This enforces network-level security by only allowing traffic from the intended instances, further isolating and securing the API calls.

D. Modify the ACL on the endpoint network interface to restrict access to the instances.

Network ACLs (Access Control Lists) are stateless firewalls that operate at the subnet level. While they can provide a basic level of security, security groups are more granular and are directly associated with network interfaces. Security groups are stateful, meaning they remember allowed connections, whereas ACLs do not. Since the goal is to restrict access to *specific instances* (rather than entire subnets), security groups offer the required precision. Furthermore, VPC endpoint documentation recommends using Security Groups for controlling traffic.

E. Add a SageMaker Runtime VPC endpoint interface to the VPC.

The question specifically mentions securing calls to the *SageMaker Service API*, not the SageMaker Runtime API. The SageMaker Service API is used for managing SageMaker resources like creating training jobs, deploying models, etc. The SageMaker Runtime API is used for invoking deployed models for inference. A separate runtime endpoint may or may not be needed depending on the application. It is not relevant in securing the management aspect of the SageMaker Service API.

In summary: The correct approach involves using both VPC endpoint policies (A) for user/role-based access control and security groups (C) for instance-level network access control to ensure that only authorized instances and IAM users can access the SageMaker Service API through the VPC endpoint.

Supporting Links:

VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
Controlling Access to Services with VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-access.html
Security Groups: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html

Question: 117

Exam Heist

An e commerce company wants to launch a new cloud-based product recommendation feature for its web application. Due to data localization regulations, any sensitive data must not leave its on-premises data center, and the product recommendation model must be trained and tested using nonsensitive data only. Data transfer to the cloud must use IPsec. The web application is hosted on premises with a PostgreSQL database that contains all the data. The company wants the data to be uploaded securely to Amazon S3 each day for model retraining.
How should a machine learning specialist meet these requirements?

- ○ Create an AWS Glue job to connect to the PostgreSQL DB instance. Ingest tables without sensitive data through an AWS Site-to-Site VPN connection directly into Amazon S3. ✓
- ○ Create an AWS Glue job to connect to the PostgreSQL DB instance. Ingest all data through an AWS Site-to-Site VPN connection into Amazon S3 while removing sensitive data using a PySpark job.
- ○ Use AWS Database Migration Service (AWS DMS) with table mapping to select PostgreSQL tables with no sensitive data through an SSL connection. Replicate data directly into Amazon S3.
- ○ Use PostgreSQL logical replication to replicate all data to PostgreSQL in Amazon EC2 through AWS Direct Connect with a VPN connection. Use AWS Glue to move data from Amazon EC2 to Amazon S3.

Explanation:

Option A is the most suitable solution because it directly addresses all requirements efficiently and securely. It avoids unnecessary data duplication and minimizes complexity.

Data Localization: The solution only transfers non-sensitive data to the cloud, adhering to data localization regulations.
Secure Transfer: It uses AWS Site-to-Site VPN, satisfying the requirement for IPsec encryption during data transfer. AWS Site-to-Site VPN establishes a secure tunnel between the on-premises network and AWS. (See: https://aws.amazon.com/vpn/site-to-site-vpn/)
Efficient Data Selection: AWS Glue is a serverless data integration service that can connect to various data sources, including PostgreSQL. Using an AWS Glue job allows the machine learning specialist to select only tables without sensitive data before transferring them to Amazon S3. This pre-selection is more efficient than transferring all data and then removing sensitive parts. (See: https://aws.amazon.com/glue/)
Direct S3 Upload: Direct ingestion into S3 simplifies the data pipeline, avoiding intermediate storage or processing steps.
Option B is less efficient as it involves transferring all data, including sensitive data, to AWS before removing the sensitive data. This approach increases the risk of sensitive data exposure and adds an unnecessary processing step.
Option C uses AWS DMS and SSL which is valid for transferring data securely, but SSL is not IPsec, violating a requirement. More importantly, AWS DMS's table mapping features are less straightforward and efficient for only selecting certain tables or a subset of data, compared to the capabilities offered by AWS Glue's data filtering and transformation capabilities.
Option D is the most complex and expensive option. It requires setting up an EC2 instance with PostgreSQL, replicating all data, and then using AWS Glue to transfer data to Amazon S3. This approach adds unnecessary infrastructure and complexity without providing any significant benefit. It also requires AWS Direct Connect in addition to VPN which is an unnecessary expense.
Therefore, Option A offers the most efficient, secure, and compliant solution for the given requirements.

Question: 118

Exam Heist

A logistics company needs a forecast model to predict next month's inventory requirements for a single item in 10 warehouses. A machine learning specialist uses
Amazon Forecast to develop a forecast model from 3 years of monthly data. There is no missing data. The specialist selects the DeepAR+ algorithm to train a predictor. The predictor means absolute percentage error (MAPE) is much larger than the MAPE produced by the current human forecasters.
Which changes to the CreatePredictor API call could improve the MAPE? (Choose two.)

- [x] Set PerformAutoML to true. ✓
- [ ] Set ForecastHorizon to 4.
- [ ] Set ForecastFrequency to W for weekly.
- [x] Set PerformHPO to true. ✓
- [ ] Set FeaturizationMethodName to filling.

Explanation:
The original DeepAR+ model is underperforming human forecasters, indicating a need for model optimization. Let's analyze the options:
A. **Set PerformAutoML to true.** *Correct.* AutoML automates the model selection and hyperparameter tuning process. By setting `PerformAutoML` to true, Amazon Forecast will automatically test different algorithms (besides DeepAR+) and hyperparameter configurations to find the best performing model for the given dataset. This could discover a more suitable algorithm or parameter set that significantly improves MAPE. [https://docs.aws.amazon.com/forecast/latest/dg/auto-ml.html]
B. **Set ForecastHorizon to 4.** *Incorrect.* The `ForecastHorizon` determines how far into the future the model predicts. Changing it to 4 doesn't address the underlying issue of inaccurate model fitting to historical data. It simply changes the length of the prediction period. It could even potentially worsen the problem if the data further out has a different distribution.
C. **Set ForecastFrequency to W for weekly.** *Incorrect.* The data is monthly. Changing the `ForecastFrequency` to weekly when the input data is monthly is inappropriate and would likely introduce significant errors. The frequency must align with the granularity of the input time series.
D. **Set PerformHPO to true.** *Correct.* Hyperparameter optimization (HPO) focuses on tuning the hyperparameters of the chosen algorithm (DeepAR+ in this case) to improve its performance. Setting `PerformHPO` to true instructs Amazon Forecast to search for optimal hyperparameter values through experimentation. Better hyperparameters can lead to a more accurate model and lower MAPE. [https://docs.aws.amazon.com/forecast/latest/dg/hyperparameter-optimization.html]
E. **Set FeaturizationMethodName to filling.** *Incorrect.* Featurization methods handle missing values in the input data. The problem states that there's no missing data, so featurization is irrelevant and `filling` won't help. Featurization involves converting raw data into features usable by the model. This is not the current problem.

Question: 119

Exam Heist

A data scientist wants to use Amazon Forecast to build a forecasting model for inventory demand for a retail company. The company has provided a dataset of historic inventory demand for its products as a .csv file stored in an Amazon S3 bucket. The table below shows a sample of the dataset.

timestamp	item_id	demand	category	lead_time
2019-12-14	uni_000736	120	hardware	90
2020-01-31	uni_003429	98	hardware	30
2020-03-04	uni_000211	234	accessories	10

How should the data scientist transform the data?

- Use ETL jobs in AWS Glue to separate the dataset into a target time series dataset and an item metadata dataset. Upload both datasets as .csv files to Amazon S3. ✓
- Use a Jupyter notebook in Amazon SageMaker to separate the dataset into a related time series dataset and an item metadata dataset. Upload both datasets as tables in Amazon Aurora.
- Use AWS Batch jobs to separate the dataset into a target time series dataset, a related time series dataset, and an item metadata dataset. Upload them directly to Forecast from a local machine.
- Use a Jupyter notebook in Amazon SageMaker to transform the data into the optimized protobuf recordIO format. Upload the dataset in this format to Amazon S3.

Explanation:
Amazon Forecast requires the input data to be separated into a target time series dataset and an item metadata dataset. The target time series dataset should include the time series data that you want to use for forecasting, such as inventory demand in this case. The item metadata dataset should include the metadata that describes the items in the time series, such as product IDs, categories, and attributes. Therefore, the data scientist should use ETL jobs in AWS Glue to separate the dataset into a target time series dataset and an item metadata dataset. Both datasets should be uploaded as .csv files to Amazon S3, which is a suitable storage option for input data to Amazon Forecast.

Question: 120

Exam Heist

A machine learning specialist is running an Amazon SageMaker endpoint using the built-in object detection algorithm on a P3 instance for real-time predictions in a company's production application. When evaluating the model's resource utilization, the specialist notices that the model is using only a fraction of the GPU.
Which architecture changes would ensure that provisioned resources are being utilized effectively?

- Redeploy the model as a batch transform job on an M5 instance.
- Redeploy the model on an M5 instance. Attach Amazon Elastic Inference to the instance. ✓
- Redeploy the model on a P3dn instance.
- Deploy the model onto an Amazon Elastic Container Service (Amazon ECS) cluster using a P3 instance.

Show Explanation

Question: 121

Exam Heist

A data scientist uses an Amazon SageMaker notebook instance to conduct data exploration and analysis. This requires certain Python packages that are not natively available on Amazon SageMaker to be installed on the notebook instance.
How can a machine learning specialist ensure that required packages are automatically available on the notebook instance for the data scientist to use?

- Install AWS Systems Manager Agent on the underlying Amazon EC2 instance and use Systems Manager Automation to execute the package installation commands.
- Create a Jupyter notebook file (.ipynb) with cells containing the package installation commands to execute and place the file under the /etc/init directory of each Amazon SageMaker notebook instance.
- Use the conda package manager from within the Jupyter notebook console to apply the necessary conda packages to the default kernel of the notebook.
- Create an Amazon SageMaker lifecycle configuration with package installation commands and assign the lifecycle configuration to the notebook instance. ✓

Explanation:
The correct answer is D: Create an Amazon SageMaker lifecycle configuration with package installation commands and assign the lifecycle configuration to the notebook instance. Here's why:

Lifecycle configurations in Amazon SageMaker provide a mechanism to automate customization of notebook instances. They are shell scripts that run when a notebook instance is created or when it's started. This makes them ideal for installing packages, configuring environments, and performing other setup tasks that ensure the notebook instance is ready to use with the necessary dependencies. The data scientist doesn't need to manually install the packages each time the notebook is created or restarted.
Option A is incorrect because while AWS Systems Manager could be used to manage EC2 instances, it's an overcomplicated solution for this specific need. Lifecycle configurations are specifically designed for SageMaker notebook instance setup. Systems Manager also requires more configuration and setup to manage instances, including IAM roles and SSM agent setup.
Option B is incorrect because putting commands directly in `/etc/init` might not be the right approach for SageMaker notebook instances and could lead to unexpected behavior or conflicts with SageMaker's internal processes. Lifecycle configurations are the recommended and supported way to customize these instances. While placing script within /etc/init directory would technically be executed, it can lead to system instability.
Option C is incorrect because while `conda` can install packages, it requires the data scientist to manually run the commands each time, defeating the goal of automation. The aim is to have the packages automatically installed whenever the instance starts. A lifecycle configuration automates this.
Using lifecycle configurations offers several advantages:

Automation: Packages are installed automatically upon instance creation or startup.
Consistency: Ensures all notebook instances have the same required packages.
Centralized management: Configuration is defined and managed in a single location.
Ease of use: Simplifies the setup process for data scientists.
Scalability: Easily applied to multiple notebook instances.
Therefore, using a lifecycle configuration is the best approach to ensure the required packages are automatically available for the data scientist's use on the SageMaker notebook instance.
Relevant Documentation:

Amazon SageMaker Lifecycle Configurations

Question: 122

Exam Heist

A data scientist needs to identify fraudulent user accounts for a company's ecommerce platform. The company wants the ability to determine if a newly created account is associated with a previously known fraudulent user. The data scientist is using AWS Glue to cleanse the company's application logs during ingestion.
Which strategy will allow the data scientist to identify fraudulent accounts?

- Execute the built-in FindDuplicates Amazon Athena query.
- Create a FindMatches machine learning transform in AWS Glue. ✓
- Create an AWS Glue crawler to infer duplicate accounts in the source data.
- Search for duplicate accounts in the AWS Glue Data Catalog.

Explanation:
The correct answer is B: Create a FindMatches machine learning transform in AWS Glue.
Here's why: The goal is to identify potentially fraudulent accounts by linking new accounts to previously known fraudulent users based on similarities in data. AWS Glue's FindMatches ML Transform is specifically designed for record linkage and deduplication tasks, which is perfectly suited for this scenario. It leverages machine learning algorithms to identify records that refer to the same entity, even if they have variations in the data (e.g., slightly different names, addresses, or email addresses). This is crucial because fraudulent actors often use variations of their information to create new accounts.
Option A (Execute the built-in FindDuplicates Amazon Athena query) is not ideal because simple duplicate queries usually look for exact matches. Fraudulent users rarely create exact duplicate accounts.
Option C (Create an AWS Glue crawler to infer duplicate accounts in the source data) is incorrect because a crawler's primary function is to discover and catalog data, not to perform advanced matching or deduplication using machine learning. Crawlers infer the schema of the data but do not inherently identify similar but not identical records.
Option D (Search for duplicate accounts in the AWS Glue Data Catalog) is also incorrect. The Data Catalog is a metadata repository. While it stores information about the data, it doesn't provide functionalities for matching similar records or identifying potential fraud.
The FindMatches ML Transform in AWS Glue provides algorithms to find fuzzy matches based on various criteria like Jaro-Winkler distance, cosine similarity, or custom match algorithms. It can learn from labeled data (e.g., known fraudulent accounts) to improve its accuracy over time. This approach enables the data scientist to build a robust system for detecting fraudulent accounts by identifying associations with previously known fraudulent users, even when they use slightly different information.

For further research:

AWS Glue FindMatches: https://docs.aws.amazon.com/glue/latest/dg/machine-learning-find-matches.html
AWS Glue Documentation: https://docs.aws.amazon.com/glue/index.html

Question: 123

Exam Heist

A Data Scientist is developing a machine learning model to classify whether a financial transaction is fraudulent. The labeled data available for training consists of
100,000 non-fraudulent observations and 1,000 fraudulent observations.
The Data Scientist applies the XGBoost algorithm to the data, resulting in the following confusion matrix when the trained model is applied to a previously unseen validation dataset. The accuracy of the model is 99.1%, but the Data Scientist needs to reduce the number of false negatives.

	Predicted 0	Predicted 1
Actual 0	99,966	34
Actual 1	877	123

Which combination of steps should the Data Scientist take to reduce the number of false negative predictions by the model? (Choose two.)

- [ ] Change the XGBoost eval_metric parameter to optimize based on Root Mean Square Error (RMSE).
- [x] Increase the XGBoost scale_pos_weight parameter to adjust the balance of positive and negative weights. ✓
- [ ] Increase the XGBoost max_depth parameter because the model is currently underfitting the data.
- [x] Change the XGBoost eval_metric parameter to optimize based on Area Under the ROC Curve (AUC). ✓
- [ ] Decrease the XGBoost max_depth parameter because the model is currently overfitting the data.

Explanation:
B. Increasing the XGBoost scale_pos_weight parameter will adjust the balance of positive and negative weights, which can help reduce the number of false negatives. By giving more weight to the positive class (fraudulent transactions), the model will be more sensitive to detecting them.
D. Changing the XGBoost eval_metric parameter to optimize based on Area Under the ROC Curve (AUC) will prioritize the model's ability to distinguish between classes. Maximizing the AUC will improve the model's ability to correctly classify both positive and negative cases, thus reducing false negatives.

Question: 124

Exam Heist

A data scientist has developed a machine learning translation model for English to Japanese by using Amazon SageMaker's built-in seq2seq algorithm with
500,000 aligned sentence pairs. While testing with sample sentences, the data scientist finds that the translation quality is reasonable for an example as short as five words. However, the quality becomes unacceptable if the sentence is 100 words long.
Which action will resolve the problem?

- ( ) Change preprocessing to use n-grams.
- ( ) Add more nodes to the recurrent neural network (RNN) than the largest sentence's word count.
- (x) Adjust hyperparameters related to the attention mechanism. ✓
- ( ) Choose a different weight initialization type.

Explanation:
The problem is that the translation quality degrades significantly for longer sentences. This points to a limitation in the model's ability to handle long-range dependencies between words in the input and output sequences. While sequence-to-sequence models can handle variable-length inputs and outputs, their performance can degrade for longer sequences due to the vanishing gradient problem and the inability to effectively capture long-range dependencies.
The attention mechanism addresses this limitation by allowing the decoder to focus on relevant parts of the input sequence when generating each word in the output sequence. By learning to attend to the most important parts of the input, the model can effectively handle long sequences and improve translation quality.
Option A, using n-grams, is more relevant for language modeling, not translation. Option B, adding more nodes, might slightly improve the model but does not address the core issue of handling long-range dependencies. Option D, choosing a different weight initialization, is important for training stability but won't specifically solve the long-sentence degradation issue.

Therefore, adjusting hyperparameters related to the attention mechanism is the most appropriate solution. These hyperparameters might include the attention type (e.g., Bahdanau, Luong), attention size, and regularization techniques. Tuning these parameters can help the model learn to better attend to relevant parts of the input sequence, leading to improved translation quality for longer sentences.
Relevant Research:

Attention Mechanism
Sequence-to-Sequence Learning with Neural Networks

Question: 125

Exam Heist

A financial company is trying to detect credit card fraud. The company observed that, on average, 2% of credit card transactions were fraudulent. A data scientist trained a classifier on a year's worth of credit card transactions data. The model needs to identify the fraudulent transactions (positives) from the regular ones
(negatives). The company's goal is to accurately capture as many positives as possible.
Which metrics should the data scientist use to optimize the model? (Choose two.)

- [ ] Specificity
- [ ] False positive rate
- [ ] Accuracy
- [x] Area under the precision-recall curve ✓
- [x] True positive rate ✓

Explanation:
Here's a detailed justification for why options D (Area under the precision-recall curve) and E (True positive rate) are the most appropriate metrics in this scenario, along with why the other options are less suitable.
Justification for D (Area under the Precision-Recall Curve - AUC-PR):
Because the data is imbalanced (2% fraud rate), focusing on overall accuracy (option C) can be misleading. A model that predicts *everything* as non-fraudulent would achieve high accuracy (98%) but be useless for fraud detection. Precision-Recall curves are particularly useful for imbalanced datasets. Precision (TP / TP + FP) measures the accuracy of positive predictions, while recall (TP / TP + FN) measures the ability to find all positive instances. The AUC-PR represents the average precision across different recall thresholds. A higher AUC-PR indicates a better trade-off between precision and recall, which is crucial when the cost of missing fraudulent transactions (false negatives) is high. The company desires to capture as many positives as possible. Optimizing for a high AUC-PR will help the model perform well specifically in identifying fraud, regardless of class imbalance.
Justification for E (True Positive Rate - TPR) also known as Recall:
True Positive Rate (TPR), also known as recall or sensitivity, directly measures the proportion of actual fraudulent transactions that are correctly identified by the model. In a fraud detection context, maximizing TPR is vital because missing fraudulent transactions can lead to significant financial losses and reputational damage. The question explicitly states the goal is to capture as many positives (fraudulent transactions) as possible, making TPR a direct measure of the model's success. A higher TPR means fewer fraudulent transactions are slipping through the cracks.
Why other options are less suitable:

A (Specificity): Specificity (TN / TN + FP) measures the proportion of actual non-fraudulent transactions that are correctly identified. While important, it's secondary to capturing fraud in this context. We want to catch the fraud even if it means some legitimate transactions are flagged for review.
B (False Positive Rate): False Positive Rate (FP / TN + FP) measures the proportion of actual non-fraudulent transactions that are incorrectly identified as fraudulent. Minimizing the FPR is important to reduce unnecessary investigations. However, it's not as critical as maximizing TPR when the primary goal is to catch as much fraud as possible.
C (Accuracy): As mentioned above, accuracy is a poor metric for imbalanced datasets because the high number of true negatives will dominate the score, masking poor performance on the positive (fraudulent) class.
Authoritative Links for further research:

Precision-Recall Curve: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall
Imbalanced Data: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
Recall/Sensitivity: https://en.wikipedia.org/wiki/Sensitivity_and_specificity

In summary, optimizing for a high AUC-PR and TPR prioritizes the detection of fraudulent transactions, which aligns directly with the financial company's stated goal. These metrics are more informative and appropriate than accuracy, specificity, or FPR in the context of imbalanced data and the specific business objective.

Question: 126

Exam Heist

A machine learning specialist is developing a proof of concept for government users whose primary concern is security. The specialist is using Amazon
SageMaker to train a convolutional neural network (CNN) model for a photo classifier application. The specialist wants to protect the data so that it cannot be accessed and transferred to a remote host by malicious code accidentally installed on the training container.
Which action will provide the MOST secure protection?

- Remove Amazon S3 access permissions from the SageMaker execution role.
- Encrypt the weights of the CNN model.
- Encrypt the training and validation dataset.
- Enable network isolation for training jobs. ✓

Explanation:

The most secure protection against malicious code exfiltrating data from a SageMaker training container is enabling network isolation. Network isolation restricts the training container from accessing any network resources, including the internet and other AWS services, except for explicitly permitted connections. This means even if malicious code were to somehow get onto the container, it wouldn't be able to send the training data or model artifacts to an external, potentially hostile, destination.
Option A, removing S3 access, would prevent the training job from accessing the training data in S3 and saving the model artifacts back to S3, making the training job impossible to run in the first place. While it restricts data access, it isn't a practical solution for a functional training process.
Option B, encrypting model weights, protects the model *after* it's trained and stored. It does nothing to prevent data exfiltration *during* the training process.
Option C, encrypting the training and validation datasets, secures the data at rest and in transit, but it does not prevent a malicious process within the training container from reading the decrypted data and sending it out if the container has network access.
Network isolation is the only option that actively prevents data from leaving the training environment, thus addressing the specific security concern of data exfiltration by malicious code. By effectively air-gapping the training container, network isolation provides a strong security barrier.
References:

Amazon SageMaker Security: https://docs.aws.amazon.com/sagemaker/latest/dg/security.html
Network Isolation in SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/train-vpc.html

Question: 127

Exam Heist

A medical imaging company wants to train a computer vision model to detect areas of concern on patients' CT scans. The company has a large collection of unlabeled CT scans that are linked to each patient and stored in an Amazon S3 bucket. The scans must be accessible to authorized users only. A machine learning engineer needs to build a labeling pipeline.
Which set of steps should the engineer take to build the labeling pipeline with the LEAST effort?

- Create a workforce with AWS Identity and Access Management (IAM). Build a labeling tool on Amazon EC2 Queue images for labeling by using Amazon Simple Queue Service (Amazon SQS). Write the labeling instructions.
- Create an Amazon Mechanical Turk workforce and manifest file. Create a labeling job by using the built-in image classification task type in Amazon SageMaker Ground Truth. Write the labeling instructions.
- Create a private workforce and manifest file. Create a labeling job by using the built-in bounding box task type in Amazon SageMaker Ground Truth. Write the labeling instructions. ✓
- Create a workforce with Amazon Cognito. Build a labeling web application with AWS Amplify. Build a labeling workflow backend using AWS Lambda. Write the labeling instructions.

Explanation:

The correct answer is C. Here's why:

SageMaker Ground Truth Simplifies Labeling Pipelines: Ground Truth is designed specifically for creating and managing labeling workflows. It offers built-in task types and workforce management, reducing the need for custom development. https://aws.amazon.com/sagemaker/groundtruth/
Private Workforce for Controlled Access: Given the sensitivity of medical data, a private workforce in Ground Truth aligns with the requirement of authorized access to the scans.
Bounding Box Task Type: Detecting areas of concern on CT scans directly translates to a bounding box task, where labelers draw boxes around the regions of interest. Ground Truth offers a built-in task type for this.
Manifest File for Data Input: A manifest file provides a structured way to tell Ground Truth where the unlabeled images reside in S3.
Less Effort Compared to Alternatives: Option A requires building a labeling tool on EC2 and managing a queue with SQS, which involves significantly more development effort. Option D, while possible, also necessitates custom web application development with Amplify and backend logic with Lambda, making it far more complex than Ground Truth's built-in capabilities. Option B uses Amazon Mechanical Turk, which is good for publicly available data. Private medical data should not be handled using this method.
IAM Integration: Ground Truth integrates with AWS IAM to manage access to labeling resources and data, ensuring security.
In contrast:

Option A: Building a labeling tool from scratch on EC2 is time-consuming and requires expertise in web development and queue management with SQS.
Option B: Amazon Mechanical Turk uses public workforce, which is not the ideal approach for healthcare data.
Option D: Requires building a front-end using Amplify and back-end using Lambda which requires more effort.
Therefore, using SageMaker Ground Truth with a private workforce and the bounding box task type minimizes development effort while meeting the security requirements.

Question: 128

Exam Heist

A company is using Amazon Textract to extract textual data from thousands of scanned text-heavy legal documents daily. The company uses this information to process loan applications automatically. Some of the documents fail business validation and are returned to human reviewers, who investigate the errors. This activity increases the time to process the loan applications.
What should the company do to reduce the processing time of loan applications?

- Configure Amazon Textract to route low-confidence predictions to Amazon SageMaker Ground Truth. Perform a manual review on those words before performing a business validation.
- Use an Amazon Textract synchronous operation instead of an asynchronous operation.
- Configure Amazon Textract to route low-confidence predictions to Amazon Augmented AI (Amazon A2I). Perform a manual review on those words before performing a business validation. ✓
- Use Amazon Rekognition's feature to detect text in an image to extract the data from scanned images. Use this information to process the loan applications.

Explanation:
The correct answer is **C. Configure Amazon Textract to route low-confidence predictions to Amazon Augmented AI (Amazon A2I). Perform a manual review on those words before performing a business validation.**
Here's why:
The problem lies in documents failing business validation due to errors in the extracted text, leading to human intervention and increased processing time. To address this, we need a mechanism to improve the accuracy of the extracted text *before* it reaches the business validation step.
Amazon A2I is specifically designed to integrate human review workflows into machine learning applications. By configuring Textract to send low-confidence predictions to A2I, human reviewers can correct or validate the extracted text where the model is uncertain. This pre-emptive correction minimizes the chances of documents failing downstream business validation. This approach reduces the overall loan processing time by preemptively catching and correcting errors that would otherwise lead to failures.
Let's examine why the other options are less suitable:

A. Configure Amazon Textract to route low-confidence predictions to Amazon SageMaker Ground Truth. Perform a manual review on those words before performing a business validation. While Ground Truth is a great tool for labeling data to *train* models, it's less efficient and cost-effective for ongoing human review of individual predictions in a production setting. A2I is specifically built for this kind of operational workflow. Ground Truth's primary goal is to create training datasets, whereas A2I's is to facilitate human-in-the-loop workflows for active ML applications.
B. Use an Amazon Textract synchronous operation instead of an asynchronous operation. Switching to a synchronous operation might reduce the *latency* of the initial text extraction, but it doesn't address the core issue of *accuracy*. If the extracted text is still incorrect, the documents will still fail business validation. Synchronous operations are also more suitable

for smaller batches of documents, not "thousands daily" as described in the scenario.
D. Use Amazon Rekognition's feature to detect text in an image to extract the data from scanned images. Use this information to process the loan applications. Amazon Rekognition is designed for object and scene detection and image analysis. While it *can* detect text in images, it is not optimized for extracting structured textual data from documents in the way that Textract is. Textract is specialized for OCR and document understanding. It is significantly better at maintaining the document structure than Rekognition.
In summary, Amazon A2I offers the most efficient and purpose-built solution for integrating human review into the Textract workflow, minimizing errors, and reducing the overall loan processing time.
Relevant Resources:

Amazon Augmented AI (A2I): https://aws.amazon.com/augmented-ai/
Amazon Textract: https://aws.amazon.com/textract/
Amazon SageMaker Ground Truth: https://aws.amazon.com/sagemaker/groundtruth/
Amazon Rekognition: https://aws.amazon.com/rekognition/

Question: 129

Exam Heist

A company ingests machine learning (ML) data from web advertising clicks into an Amazon S3 data lake. Click data is added to an Amazon Kinesis data stream by using the Kinesis Producer Library (KPL). The data is loaded into the S3 data lake from the data stream by using an Amazon Kinesis Data Firehose delivery stream. As the data volume increases, an ML specialist notices that the rate of data ingested into Amazon S3 is relatively constant. There also is an increasing backlog of data for Kinesis Data Streams and Kinesis Data Firehose to ingest.
Which next step is MOST likely to improve the data ingestion rate into Amazon S3?

- Increase the number of S3 prefixes for the delivery stream to write to.
- Decrease the retention period for the data stream.
- Increase the number of shards for the data stream. ✓
- Add more consumers using the Kinesis Client Library (KCL).

Explanation:
The problem is that the data ingestion rate into S3 is constant despite increasing data volume, causing a backlog in Kinesis Data Streams and Kinesis Data Firehose. This indicates a bottleneck in processing the stream data.
Option C, increasing the number of shards for the data stream, directly addresses this bottleneck. Kinesis Data Streams uses shards as the fundamental unit of throughput. Each shard provides a fixed write and read capacity. By increasing the number of shards, the data stream's overall capacity to ingest and process data is increased, allowing Kinesis Data Firehose to consume data faster and subsequently write more data to S3. More shards mean more parallel processing of the data stream.
Option A, increasing the number of S3 prefixes, might improve write performance to S3 to some extent by reducing contention on individual S3 prefixes, but it won't address the fundamental bottleneck in the Kinesis Data Streams. S3 prefixes come into play *after* the data has been successfully delivered to Kinesis Data Firehose.
Option B, decreasing the retention period, only impacts how long data is stored in Kinesis Data Streams. It doesn't increase the ingestion or processing rate. The problem isn't how long the data is kept, but that it isn't being processed fast enough.
Option D, adding more consumers using KCL, is relevant when there's a need to perform multiple, independent processing tasks on the same data stream. In this scenario, Kinesis Data Firehose is *already* acting as a consumer. Adding *more* consumers *besides* Kinesis Data Firehose is outside the scope of the specified architecture and wouldn't solve the problem of the Firehose bottleneck. Moreover, adding KCL consumers could potentially exacerbate the problem by further straining the stream's capacity, if the KCL consumers are not efficient.
Therefore, increasing the number of shards (Option C) is the most direct and effective way to increase the data ingestion rate into Amazon S3 by alleviating the bottleneck in Kinesis Data Streams and allowing Kinesis Data Firehose to operate more efficiently.
Here are authoritative links for further reading:

Amazon Kinesis Data Streams Scalability: https://docs.aws.amazon.com/streams/latest/dev/kinesis-scaling.html
Amazon Kinesis Data Firehose Limits: https://docs.aws.amazon.com/firehose/latest/dev/limits.html

Question: 130

Exam Heist

A data scientist must build a custom recommendation model in Amazon SageMaker for an online retail company. Due to the nature of the company's products, customers buy only 4-5 products every 5-10 years. So, the company relies on a steady stream of new customers. When a new customer signs up, the company collects data on the customer's preferences. Below is a sample of the data available to the data scientist.

timestamp	user_id	product_id	preference_1	...	preference_10
2020-03-04	90	25	0	...	0.374
2020-03-04	90	61	0	...	0.374
2020-02-21	203	56	1	...	0.098

How should the data scientist split the dataset into a training and test set for this use case?

- Shuffle all interaction data. Split off the last 10% of the interaction data for the test set.
- Identify the most recent 10% of interactions for each user. Split off these interactions for the test set. ✓
- Identify the 10% of users with the least interaction data. Split off all interaction data from these users for the test set.
- Randomly select 10% of the users. Split off all interaction data from these users for the test set.

Explanation:
This method is appropriate because it takes into account the unique buying behaviour of each customer and is likely to reflect the latest preferences of the customer. It ensures that the test set contains a representative sample of the most recent customer preferences, which is important in this use case where customer preferences change infrequently over time. B makes the most business sense. Since customers buy products every 4-5 years, it makes sense to be able to predict future sales from really old data. splitting the test set to be only recent interactions is the best way to test model performance from historically 'recent' data.

Question: 131

Exam Heist

A financial services company wants to adopt Amazon SageMaker as its default data science environment. The company's data scientists run machine learning
(ML) models on confidential financial data. The company is worried about data egress and wants an ML engineer to secure the environment. Which mechanisms can the ML engineer use to control data egress from SageMaker? (Choose three.)

- [x] Connect to SageMaker by using a VPC interface endpoint powered by AWS PrivateLink. ✓
- [ ] Use SCPs to restrict access to SageMaker.
- [ ] Disable root access on the SageMaker notebook instances.
- [x] Enable network isolation for training jobs and models. ✓
- [x] Restrict notebook presigned URLs to specific IPs used by the company. ✓
- [ ] Protect data with encryption at rest and in transit. Use AWS Key Management Service (AWS KMS) to manage encryption keys.

Explanation:
The correct answer is ADE. Here's a breakdown of why each option is either right or wrong, focusing on controlling data egress from SageMaker:

A. Connect to SageMaker by using a VPC interface endpoint powered by AWS PrivateLink. This is correct. AWS PrivateLink allows you to connect to SageMaker within your VPC without exposing traffic to the public internet. All traffic stays within the AWS network, preventing data from leaving your controlled environment.
https://docs.aws.amazon.com/sagemaker/latest/dg/interface-vpc-endpoint.html
B. Use SCPs to restrict access to SageMaker. While Service Control Policies (SCPs) are useful for managing permissions at an organizational level, they primarily restrict *who* can access SageMaker resources, not necessarily *how* data can egress from already-accessible resources. SCPs control account-level permissions, not data flow within SageMaker.
C. Disable root access on the SageMaker notebook instances. While restricting root access is a good security practice, it primarily focuses on preventing unauthorized modifications to the instance itself and does not directly prevent data from being copied out of the environment by an authorized user with sufficient permissions.
D. Enable network isolation for training jobs and models. This is correct. Enabling network isolation ensures that training jobs and model endpoints do not have access to the internet by default. This prevents them from sending data outside the VPC.
https://docs.aws.amazon.com/sagemaker/latest/dg/train-vpc.html and
https://docs.aws.amazon.com/sagemaker/latest/dg/model-vpc.html
E. Restrict notebook presigned URLs to specific IPs used by the company. This is correct. Presigned URLs can grant temporary access to data. Limiting access to only known company IP addresses ensures that only users within the company network can access the data through these URLs, preventing external unauthorized access.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-presigned-urls.html
F. Protect data with encryption at rest and in transit. Use AWS Key Management Service (AWS KMS) to manage encryption keys. While encryption is crucial for data security, it primarily addresses data confidentiality, not data egress. Encryption protects data from being read if it is improperly accessed, but it doesn't prevent authorized users within the environment from

copying the encrypted data and potentially decrypting it elsewhere if they have the necessary keys.

Question: 132

Exam Heist

A company is converting a large number of unstructured paper receipts into images. The company wants to create a model based on natural language processing
(NLP) to find relevant entities such as date, location, and notes, as well as some custom entities such as receipt numbers.
The company is using optical character recognition (OCR) to extract text for data labeling. However, documents are in different structures and formats, and the company is facing challenges with setting up the manual workflows for each document type. Additionally, the company trained a named entity recognition (NER) model for custom entity detection using a small sample size. This model has a very low confidence score and will require retraining with a large dataset.
Which solution for text extraction and entity detection will require the LEAST amount of effort?

- Extract text from receipt images by using Amazon Textract. Use the Amazon SageMaker BlazingText algorithm to train on the text for entities and custom entities.
- Extract text from receipt images by using a deep learning OCR model from the AWS Marketplace. Use the NER deep learning model to extract entities.
- Extract text from receipt images by using Amazon Textract. Use Amazon Comprehend for entity detection, and use Amazon Comprehend custom entity recognition for custom entity detection. ✓
- Extract text from receipt images by using a deep learning OCR model from the AWS Marketplace. Use Amazon Comprehend for entity detection, and use Amazon Comprehend custom entity recognition for custom entity detection.

Explanation:
The correct answer is **C**. Here's why:

Text Extraction: Amazon Textract is purpose-built for extracting text and structured data from documents like receipts. It is a managed service designed to automate the extraction of text, handwriting, and data from scanned documents. A deep learning OCR model from AWS Marketplace (options B and D) may require more configuration and management.
Entity Detection: Amazon Comprehend is a fully managed natural language processing (NLP) service for detecting entities, key phrases, sentiments, and more. It handles the complexities of NLP model training and deployment. For standard entities (date, location, etc.), Comprehend has pre-trained models, minimizing effort.
Custom Entity Recognition: Amazon Comprehend Custom Entity Recognition allows you to train Comprehend to recognize entities specific to your business or industry. The company already identified the need for custom entity recognition for "receipt numbers". Using Comprehend custom entity recognition enables to train a custom model specifically for this purpose.
Least Effort: Option C requires the least amount of effort because it leverages managed AWS services that are specifically designed for text extraction and entity recognition. It avoids the overhead of managing custom OCR models from AWS Marketplace and training custom NER models with SageMaker BlazingText, both of which involve significantly more effort. BlazingText (option A) is good for word embeddings and text classification but not ideal for out-of-the-box NER compared to comprehend. Option B and D use AWS Marketplace, which requires more configuration and management.
In summary, Textract handles text extraction, Comprehend handles standard entities, and Comprehend Custom Entity Recognition addresses custom entities efficiently using pre-built functionality and minimizing manual configuration.
Supporting links:

Amazon Textract: https://aws.amazon.com/textract/
Amazon Comprehend: https://aws.amazon.com/comprehend/
Amazon Comprehend Custom Entity Recognition: https://docs.aws.amazon.com/comprehend/latest/dg/cer.html

Question: 133

Exam Heist

A company is building a predictive maintenance model based on machine learning (ML). The data is stored in a fully private Amazon S3 bucket that is encrypted at rest with AWS Key Management Service (AWS KMS) CMKs. An ML specialist must run data preprocessing by using an Amazon SageMaker Processing job that is triggered from code in an Amazon SageMaker notebook. The job should read data from Amazon S3, process it, and upload it back to the same S3 bucket.
The preprocessing code is stored in a container image in Amazon Elastic Container Registry (Amazon ECR). The ML specialist needs to grant permissions to ensure a smooth data preprocessing workflow.
Which set of actions should the ML specialist take to meet these requirements?

- Create an IAM role that has permissions to create Amazon SageMaker Processing jobs, S3 read and write access to the relevant S3 bucket, and appropriate KMS and ECR permissions. Attach the role to the SageMaker notebook instance. Create an Amazon SageMaker Processing job from the notebook. ✓

- Create an IAM role that has permissions to create Amazon SageMaker Processing jobs. Attach the role to the SageMaker notebook instance. Create an Amazon SageMaker Processing job with an IAM role that has read and write permissions to the relevant S3 bucket, and appropriate KMS and ECR permissions.
- Create an IAM role that has permissions to create Amazon SageMaker Processing jobs and to access Amazon ECR.Attach the role to the SageMaker notebook instance. Set up both an S3 endpoint and a KMS endpoint in the default VPC. Create Amazon SageMaker Processing jobs from the notebook.
- Create an IAM role that has permissions to create Amazon SageMaker Processing jobs. Attach the role to the SageMaker notebook instance. Set up an S3 endpoint in the default VPC. Create Amazon SageMaker Processing jobs with the access key and secret key of the IAM user with appropriate KMS and ECR permissions.

Explanation:
The correct answer is A because it embodies the principle of least privilege and leverages IAM roles effectively for secure data processing. Here's a breakdown:

IAM Role for Notebook Instance: Attaching an IAM role to the SageMaker notebook instance grants the notebook environment the necessary permissions to perform actions like creating processing jobs.
Permissions in the IAM Role: This role must include permissions for:

`sagemaker:CreateProcessingJob`: Allows the notebook to initiate preprocessing jobs.
`s3:GetObject`, `s3:PutObject`: Grants read and write access to the specified S3 bucket for data access.
`kms:Decrypt`, `kms:Encrypt`: Enables the processing job to decrypt and encrypt data using the KMS CMK associated with the S3 bucket.
`ecr:GetAuthorizationToken`, `ecr:BatchCheckLayerAvailability`, `ecr:GetDownloadUrlForLayer`, `ecr:BatchGetImage`: Permits the notebook to pull the container image from ECR.
Processing Job Execution: When the SageMaker Processing job is created, it inherits the permissions of the IAM role attached to the notebook instance, allowing it to access data, decrypt/encrypt with KMS, and pull the container image.
Security Best Practices: This approach avoids hardcoding credentials (like access keys and secret keys) directly in the notebook, which is a significant security risk. It enforces least privilege by giving the notebook instance only the permissions it absolutely needs.
Option B is incorrect because the SageMaker Processing job does not get its own dedicated IAM role. It inherits the permissions of the role attached to the notebook instance.
Option C, while suggesting an IAM role for the notebook, recommends setting up S3 and KMS endpoints, implying VPC configuration which is not explicitly mentioned as a requirement in the prompt. Endpoints add network complexity without a clear necessity given the other permissions are sufficient.
Option D is incorrect as the access key and secret key of the IAM user should not be used for the notebook instance. This is a security risk and defeats the purpose of using IAM roles.
Relevant Links:

IAM Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
SageMaker Roles: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html
KMS Encryption: https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html
VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html

Question: 134

Exam Heist

A data scientist has been running an Amazon SageMaker notebook instance for a few weeks. During this time, a new version of Jupyter Notebook was released along with additional software updates. The security team mandates that all running SageMaker notebook instances use the latest security and software updates provided by SageMaker.
How can the data scientist meet this requirements?

- Call the CreateNotebookInstanceLifecycleConfig API operation
- Create a new SageMaker notebook instance and mount the Amazon Elastic Block Store (Amazon EBS) volume from the original instance
- Stop and then restart the SageMaker notebook instance ✓
- Call the UpdateNotebookInstanceLifecycleConfig API operation

Explanation:
The correct answer is **C. Stop and then restart the SageMaker notebook instance.**
Here's a detailed justification:

Amazon SageMaker notebook instances are built on top of Amazon Linux and receive regular security and software updates. These updates are applied automatically by AWS; however, the updates only fully take effect upon a restart of the notebook instance. When a new Jupyter Notebook version or other software update becomes available for the underlying Amazon Linux operating system, simply stopping and restarting the notebook instance triggers the installation and application of those pending updates.
Option A is incorrect because the `CreateNotebookInstanceLifecycleConfig` API operation is used to customize the notebook instance's environment upon creation or start-up. While lifecycle configurations can install specific packages, they are not the primary method for applying general security and software updates.
Option B is incorrect because creating a new instance and mounting the old EBS volume will transfer the *data* stored on the volume, but it will not inherently update the operating system or software versions. The new instance will still run the older versions until a restart is performed. Furthermore, mounting an EBS volume from one instance to another is not a standard process for updating software.
Option D is incorrect because the `UpdateNotebookInstanceLifecycleConfig` API operation, like its `Create` counterpart, is used to change the scripts that run on start-up, not to force system-wide software updates. These updates are part of the managed environment handled by SageMaker when the instance restarts.
Therefore, the simplest and most direct way to apply the latest security and software updates to a running SageMaker notebook instance, as mandated by the security team, is to stop and then restart the instance. This ensures that the latest updates available from AWS are applied, bringing the instance in line with the security requirements.
Further resources:

SageMaker Notebook Instances: Official AWS documentation on SageMaker notebook instances, including information on updates and maintenance.
Lifecycle Configurations: Information about customizing the notebook instance's environment at startup, clarifying the role of Lifecycle configurations.

Question: 135 Exam Heist

A library is developing an automatic book-borrowing system that uses Amazon Rekognition. Images of library members' faces are stored in an Amazon S3 bucket.
When members borrow books, the Amazon Rekognition CompareFaces API operation compares real faces against the stored faces in Amazon S3. The library needs to improve security by making sure that images are encrypted at rest. Also, when the images are used with Amazon Rekognition. they need to be encrypted in transit. The library also must ensure that the images are not used to improve Amazon Rekognition as a service.
How should a machine learning specialist architect the solution to satisfy these requirements?

- Enable server-side encryption on the S3 bucket. Submit an AWS Support ticket to opt out of allowing images to be used for improving the service, and follow the process provided by AWS Support. ✓
- Switch to using an Amazon Rekognition collection to store the images. Use the IndexFaces and SearchFacesByImage API operations instead of the CompareFaces API operation.
- Switch to using the AWS GovCloud (US) Region for Amazon S3 to store images and for Amazon Rekognition to compare faces. Set up a VPN connection and only call the Amazon Rekognition API operations through the VPN.
- Enable client-side encryption on the S3 bucket. Set up a VPN connection and only call the Amazon Rekognition API operations through the VPN.

Explanation:
The correct answer is A because it directly addresses all the requirements with the most appropriate solutions within the AWS ecosystem.
Enabling server-side encryption on the S3 bucket (SSE-S3, SSE-KMS, or SSE-C) ensures that the images are encrypted at rest. This protects the data from unauthorized access if the storage media is compromised. SSE-S3 is often the simplest option if specific key management is not needed. SSE-KMS provides the benefit of key rotation and auditability via CloudTrail. SSE-C provides complete control of the encryption keys. [https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html]
Amazon Rekognition, by default, might use your data to improve the service. To prevent this, AWS requires customers to explicitly opt-out. The official documentation states the process involves contacting AWS Support. While direct API control would be ideal, AWS has designed the service to use an opt-out mechanism through a support ticket to manage these requests. This ensures they can verify the specific customer's request and prevent unintended opt-outs.
Option B is incorrect because switching to Rekognition collections doesn't inherently address the encryption or data usage concerns. It only changes the way faces are stored and compared.
Option C is incorrect because while AWS GovCloud (US) provides higher security and compliance standards, it doesn't automatically address the data usage concern. Also, setting up and maintaining a VPN for all Rekognition calls is overly complex and unnecessary for encryption in transit which is already guaranteed. Traffic between AWS services is encrypted in

transit by default using TLS/SSL. Furthermore, GovCloud is designed for specific regulated workloads and using it unnecessarily can introduce complexity and cost.
Option D is incorrect because client-side encryption requires managing the encryption keys, adding operational overhead and complexity. The requirement can be more easily met with SSE. VPN is unnecessary as encryption in transit is already handled by default.

Question: 136

Exam Heist

A company is building a line-counting application for use in a quick-service restaurant. The company wants to use video cameras pointed at the line of customers at a given register to measure how many people are in line and deliver notifications to managers if the line grows too long. The restaurant locations have limited bandwidth for connections to external services and cannot accommodate multiple video streams without impacting other operations.
Which solution should a machine learning specialist implement to meet these requirements?

- Install cameras compatible with Amazon Kinesis Video Streams to stream the data to AWS over the restaurant's existing internet connection. Write an AWS Lambda function to take an image and send it to Amazon Rekognition to count the number of faces in the image. Send an Amazon Simple Notification Service (Amazon SNS) notification if the line is too long.
- Deploy AWS DeepLens cameras in the restaurant to capture video. Enable Amazon Rekognition on the AWS DeepLens device, and use it to trigger a local AWS Lambda function when a person is recognized. Use the Lambda function to send an Amazon Simple Notification Service (Amazon SNS) notification if the line is too long.
- Build a custom model in Amazon SageMaker to recognize the number of people in an image. Install cameras compatible with Amazon Kinesis Video Streams in the restaurant. Write an AWS Lambda function to take an image. Use the SageMaker endpoint to call the model to count people. Send an Amazon Simple Notification Service (Amazon SNS) notification if the line is too long.
- Build a custom model in Amazon SageMaker to recognize the number of people in an image. Deploy AWS DeepLens cameras in the restaurant. Deploy the model to the cameras. Deploy an AWS Lambda function to the cameras to use the model to count people and send an Amazon Simple Notification Service (Amazon SNS) notification if the line is too long. ✓

Explanation:
The correct answer is **D**, which prioritizes edge computing with AWS DeepLens and minimizes bandwidth usage, aligning perfectly with the restaurant's limited network capacity. Here's a detailed justification:
Bandwidth Constraint: The question explicitly states limited bandwidth, disqualifying options that rely heavily on streaming video to the cloud for processing (like option A and C using Kinesis Video Streams).
Edge Computing Advantage: AWS DeepLens enables on-device processing, reducing the need to transmit large video streams over the network. This directly addresses the bandwidth limitation.
Custom Model Necessity: The application requires counting *people*, not just recognizing faces. Amazon Rekognition (as used in A and B) offers face detection but not necessarily a reliable person count in a potentially crowded line. A custom model is needed for this specific counting task.
SageMaker for Model Building: Amazon SageMaker provides the necessary tools and infrastructure to build, train, and deploy custom machine learning models.
Model Deployment to DeepLens: After building the model in SageMaker, it can be deployed directly to the AWS DeepLens device. This allows the counting to happen locally.
Local Lambda for Notification: The AWS Lambda function deployed to DeepLens acts as an agent, using the deployed model to count people, determining if the line exceeds the threshold, and triggering the SNS notification only when necessary. This minimizes unnecessary network traffic.
Cost-Effectiveness: Edge processing reduces the cost of transmitting and processing video data in the cloud.
Real-Time Performance: On-device processing can result in faster response times since the processing is done locally without relying on network latency to a cloud service.

In summary, answer D provides a complete solution using SageMaker for custom model building, DeepLens for edge processing, and Lambda for local decision-making, optimizing for limited bandwidth and providing a real-time, cost-effective line-counting application. Options A, B, and C rely too heavily on cloud-based video streaming and lack the specificity of a custom model, rendering them less suitable given the constraints.

AWS DeepLens: https://aws.amazon.com/deeplens/
Amazon SageMaker: https://aws.amazon.com/sagemaker/
AWS Lambda: https://aws.amazon.com/lambda/
Amazon SNS: https://aws.amazon.com/sns/

Question: 137

Exam Heist

A company has set up and deployed its machine learning (ML) model into production with an endpoint using Amazon SageMaker hosting services. The ML team has configured automatic scaling for its SageMaker instances to support workload changes. During testing, the team notices that additional instances are being launched before the new instances are ready. This behavior needs to change as soon as possible.
How can the ML team solve this issue?

- Decrease the cooldown period for the scale-in activity. Increase the configured maximum capacity of instances.
- Replace the current endpoint with a multi-model endpoint using SageMaker.
- Set up Amazon API Gateway and AWS Lambda to trigger the SageMaker inference endpoint.
- Increase the cooldown period for the scale-out activity. ✓

Explanation:
The issue is that SageMaker instances are being launched too quickly during scale-out, before the newly launched instances are ready to serve traffic. This indicates that the automatic scaling configuration is triggering new instances before the previous ones have fully initialized and become operational. The desired outcome is to prevent premature instance launches during scaling.
Option D suggests increasing the cooldown period for scale-out activity. The cooldown period is the amount of time that must elapse after a scaling activity completes before another scaling activity can start. By increasing this period, the scaling process will wait longer after launching new instances before considering whether to launch even more. This allows the recently launched instances sufficient time to initialize, load the model, and become ready to handle inference requests. This addresses the core problem by preventing overly aggressive scaling.
Option A is incorrect because decreasing the scale-in cooldown would cause instances to be terminated more quickly, which is not relevant to the problem. Increasing the maximum capacity doesn't prevent premature launches; it only sets a higher upper bound for the number of instances.
Option B, using a multi-model endpoint, might improve resource utilization but doesn't directly address the timing issue of premature instance launches during scaling. Multi-model endpoints are better suited for serving multiple models from a single endpoint, not for controlling the scaling speed.
Option C, using API Gateway and Lambda, adds complexity without solving the root cause. While these services can be used in conjunction with SageMaker, they don't control the scaling behavior of the SageMaker instances. The problem lies within the automatic scaling configuration of SageMaker itself, not the way the endpoint is accessed.
Therefore, increasing the cooldown period for scale-out activity (Option D) is the most direct and effective solution. It allows the launched instances to become ready before subsequent scaling decisions are made, thus resolving the issue of launching instances prematurely.
Reference link for further research:https://docs.aws.amazon.com/autoscaling/ec2/userguide/scaling_cooldown.html

Question: 138

Exam Heist

A telecommunications company is developing a mobile app for its customers. The company is using an Amazon SageMaker hosted endpoint for machine learning model inferences.
Developers want to introduce a new version of the model for a limited number of users who subscribed to a preview feature of the app. After the new version of the model is tested as a preview, developers will evaluate its accuracy. If a new version of the model has better accuracy, developers need to be able to gradually release the new version for all users over a fixed period of time.
How can the company implement the testing model with the LEAST amount of operational overhead?

- Update the ProductionVariant data type with the new version of the model by using the CreateEndpointConfig operation with the InitialVariantWeight parameter set to 0. Specify the TargetVariant parameter for InvokeEndpoint calls for users who subscribed to the preview feature. When the new version of the model is ready for release, gradually increase InitialVariantWeight until all users have the updated version.
- Configure two SageMaker hosted endpoints that serve the different versions of the model. Create an Application Load Balancer (ALB) to route traffic to both endpoints based on the TargetVariant query string parameter. Reconfigure the app to send the TargetVariant query string parameter for users who subscribed to the preview feature. When the new version of the model is ready for release, change the ALB's routing algorithm to weighted until all users have the updated version.
- Update the DesiredWeightsAndCapacity data type with the new version of the model by using the UpdateEndpointWeightsAndCapacities operation with the DesiredWeight parameter set to 0. Specify the TargetVariant parameter for InvokeEndpoint calls for users who subscribed to the preview feature. When the new version of the model is ready for release, gradually increase DesiredWeight until all users have the updated version. ✓
- Configure two SageMaker hosted endpoints that serve the different versions of the model. Create an Amazon Route 53 record that is configured with a simple routing policy and that points to the current version of the model. Configure the mobile app to use the endpoint URL for users who subscribed to the preview feature and to use the Route 53 record for other users. When the new version of the model is ready for release, add a new model version endpoint to Route 53, and switch the policy to weighted until all users have the updated version.

Explanation:

The correct answer is C. Here's why:
Option C leverages SageMaker's built-in A/B testing capabilities with minimal operational overhead. `UpdateEndpointWeightsAndCapacities` allows you to dynamically adjust the traffic distribution between different model versions (variants) within the *same* endpoint. By setting the `DesiredWeight` of the new model to 0 initially, you ensure no traffic goes to it by default. The `TargetVariant` parameter in the `InvokeEndpoint` call enables routing preview users specifically to the new model. Gradually increasing `DesiredWeight` allows for a controlled rollout. This approach avoids the complexity of managing multiple endpoints and load balancers.
Option A is similar, but `CreateEndpointConfig` is for *creating* an endpoint configuration, not updating weights. Updating weights should be done with `UpdateEndpointWeightsAndCapacities`.
Option B introduces significant operational overhead. It requires managing two separate SageMaker endpoints and an Application Load Balancer (ALB). While ALBs can route traffic based on query parameters, this adds unnecessary complexity compared to SageMaker's built-in features. Also, reconfiguring the app to send query parameters would be cumbersome.
Option D is also overly complex. Using Route 53 for this purpose is not the best approach because Route 53 is primarily a DNS service, not a traffic management solution for A/B testing within a machine learning model. Also managing two endpoints when SageMaker has native functionality to handle model versions in one endpoint is not the ideal solution.
In summary, option C is the most efficient because it utilizes SageMaker's built-in A/B testing capabilities with minimal operational overhead, allowing for controlled release of the new model version without managing additional infrastructure.
Supporting Links:

SageMaker Endpoint Updates: https://docs.aws.amazon.com/sagemaker/latest/dg/model-ab-testing.html
UpdateEndpointWeightsAndCapacities API:
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_UpdateEndpointWeightsAndCapacities.html

Question: 139

Exam Heist

A company offers an online shopping service to its customers. The company wants to enhance the site's security by requesting additional information when customers access the site from locations that are different from their normal location. The company wants to update the process to call a machine learning (ML) model to determine when additional information should be requested.
The company has several terabytes of data from its existing ecommerce web servers containing the source IP addresses for each request made to the web server. For authenticated requests, the records also contain the login name of the requesting user.
Which approach should an ML specialist take to implement the new security feature in the web application?

- ○ Use Amazon SageMaker Ground Truth to label each record as either a successful or failed access attempt. Use Amazon SageMaker to train a binary classification model using the factorization machines (FM) algorithm.
- ○ Use Amazon SageMaker to train a model using the IP Insights algorithm. Schedule updates and retraining of the model using new log data nightly. ✓
- ○ Use Amazon SageMaker Ground Truth to label each record as either a successful or failed access attempt. Use Amazon SageMaker to train a binary classification model using the IP Insights algorithm.
- ○ Use Amazon SageMaker to train a model using the Object2Vec algorithm. Schedule updates and retraining of the model using new log data nightly.

Explanation:
The most suitable approach is **B. Use Amazon SageMaker to train a model using the IP Insights algorithm. Schedule updates and retraining of the model using new log data nightly.**
Here's why:

IP Insights Algorithm: Amazon SageMaker's IP Insights algorithm is specifically designed for anomaly detection based on IP address usage. It learns patterns of IP address associations and can identify unusual or anomalous IPs, making it ideal for detecting logins from unfamiliar locations. https://docs.aws.amazon.com/sagemaker/latest/dg/ip-insights.html
Unsupervised Learning: IP Insights is an unsupervised learning algorithm. This is crucial because the company doesn't have labeled data indicating "successful" or "failed" access attempts. Trying to label terabytes of historical data (as suggested in options A and C) would be extremely time-consuming and potentially inaccurate. The algorithm can inherently find outliers based on the log data provided.
Continuous Improvement: Scheduling nightly updates and retraining ensures that the model adapts to evolving patterns in user behavior and network configurations. This is critical for maintaining the effectiveness of the security feature over time.
Object2Vec is not the right approach: Object2Vec (option D) is designed for learning vector embeddings of objects, it learns relationships between the objects, not typically the right algorithm for the task.
Options A and C are less suitable because:

Manual Labeling is Impractical: Labeling terabytes of data as "successful" or "failed" attempts is a massive, potentially subjective, and time-consuming task.

Factorization Machines (FM) Limitations: While FM is a valid classification model, it might not be as effective as IP Insights at capturing the complex relationships between IP addresses and user behavior patterns for anomaly detection in this context.
IP Insights as Classification: You don't use IP Insights as a classification model. It's an unsupervised model used for anomaly detection.
In summary, IP Insights leverages existing log data effectively to identify anomalous login locations without requiring extensive manual labeling, aligns perfectly with the company's requirements, and allows for continual model improvement.

Question: 140

Exam Heist

A retail company wants to combine its customer orders with the product description data from its product catalog. The structure and format of the records in each dataset is different. A data analyst tried to use a spreadsheet to combine the datasets, but the effort resulted in duplicate records and records that were not properly combined. The company needs a solution that it can use to combine similar records from the two datasets and remove any duplicates.
Which solution will meet these requirements?

- Use an AWS Lambda function to process the data. Use two arrays to compare equal strings in the fields from the two datasets and remove any duplicates.
- Create AWS Glue crawlers for reading and populating the AWS Glue Data Catalog. Call the AWS Glue SearchTables API operation to perform a fuzzy-matching search on the two datasets, and cleanse the data accordingly.
- Create AWS Glue crawlers for reading and populating the AWS Glue Data Catalog. Use the FindMatches transform to cleanse the data. ✓
- Create an AWS Lake Formation custom transform. Run a transformation for matching products from the Lake Formation console to cleanse the data automatically.

Explanation:
The correct answer is C because AWS Glue's `FindMatches` transform is specifically designed for record de-duplication and finding matches across datasets, which perfectly addresses the problem of combining customer order data with product catalog data while removing duplicates and handling differing formats. The `FindMatches` transform leverages machine learning to learn how to match records even when they are not exactly identical, enabling fuzzy matching and handling inconsistencies in data. Glue crawlers are also required to properly extract the metadata from the datasets and ingest them into the Glue Data Catalog.
Option A is less suitable because implementing the fuzzy matching logic in a Lambda function using string comparisons and arrays would be complex, less scalable, and harder to maintain compared to using a managed service like AWS Glue with built-in matching capabilities.
Option B's suggestion to use the `SearchTables` API operation for fuzzy matching isn't the intended purpose of that API. `SearchTables` is used for discovering tables within the Glue Data Catalog, not for performing record linkage or de-duplication within the data itself.
Option D introduces AWS Lake Formation custom transforms, which are complex to configure and not necessary for this specific use case. Lake Formation is more suitable for managing access control to data lakes, not for implementing data cleansing and matching logic. Furthermore, Lake Formation doesn't have a built-in transform specifically tailored to fuzzy matching like Glue's `FindMatches`.
In summary, the combination of AWS Glue crawlers and the `FindMatches` transform provides the most efficient and purpose-built solution for addressing the retail company's data integration and de-duplication challenge.
Supporting links:

AWS Glue FindMatches: https://docs.aws.amazon.com/glue/latest/dg/find-matches.html
AWS Glue Crawlers: https://docs.aws.amazon.com/glue/latest/dg/add-crawler.html
AWS Glue Data Catalog: https://docs.aws.amazon.com/glue/latest/dg/datacatalog-intro.html

Question: 141

Exam Heist

A company provisions Amazon SageMaker notebook instances for its data science team and creates Amazon VPC interface endpoints to ensure communication between the VPC and the notebook instances. All connections to the Amazon SageMaker API are contained entirely and securely using the AWS network.
However, the data science team realizes that individuals outside the VPC can still connect to the notebook instances across the internet.
Which set of actions should the data science team take to fix the issue?

- Modify the notebook instances' security group to allow traffic only from the CIDR ranges of the VPC. Apply this security group to all of the notebook instances' VPC interfaces.
- Create an IAM policy that allows the sagemaker:CreatePresignedNotebookInstanceUrl and sagemaker:DescribeNotebookInstance actions from only the VPC endpoints. Apply this policy to all IAM users, groups, and roles used to access the notebook instances. ✓

- Add a NAT gateway to the VPC. Convert all of the subnets where the Amazon SageMaker notebook instances are hosted to private subnets. Stop and start all of the notebook instances to reassign only private IP addresses.
- Change the network ACL of the subnet the notebook is hosted in to restrict access to anyone outside the VPC.

Explanation:

The correct answer is B. Here's a detailed justification:

The problem is that users outside the VPC are able to access SageMaker notebook instances despite the use of VPC interface endpoints. VPC interface endpoints secure the *data plane* communication between the notebook instances and the SageMaker service API. However, they don't inherently restrict who can create a pre-signed URL to access a notebook instance.

Option B addresses this by focusing on the *control plane*. Creating a pre-signed URL for a SageMaker notebook instance requires specific IAM permissions, namely `sagemaker:CreatePresignedNotebookInstanceUrl`. If we can control *who* can create these URLs, we can effectively restrict access. By creating an IAM policy that allows the `sagemaker:CreatePresignedNotebookInstanceUrl` and `sagemaker:DescribeNotebookInstance` actions only from the VPC endpoints, we ensure that only users accessing the AWS environment through the VPC can generate these URLs. Applying this policy to the relevant IAM users, groups, and roles restricts the ability to create access URLs to those within the VPC. Those accessing from outside the VPC would not have the permissions to generate a valid pre-signed URL, effectively preventing them from accessing the notebook instance. `sagemaker:DescribeNotebookInstance` is needed so that the request can describe the notebook instance that the presigned url is for. Without this, access will also be denied.

Option A only restricts network access to the *instance itself* once a connection is already established via a pre-signed URL. It does not prevent the *creation* of a pre-signed URL by someone outside the VPC. It also does not prevent access if the notebook instance has public internet access, as would be expected in this situation.

Option C is incorrect. A NAT gateway is used to allow instances in private subnets to initiate outbound internet traffic but it doesn't restrict who can connect *to* the notebook instance via pre-signed URLs. Making subnets private won't resolve the problem of external users generating access URLs.

Option D is too restrictive and not the correct solution. Network ACLs are a layer of security that acts as a firewall for controlling traffic in and out of a subnet. Restricting access to the subnet from outside the VPC will prevent the creation of any presigned URLs which is not the best approach.

Therefore, restricting the IAM permissions to generate pre-signed URLs based on the source VPC endpoint is the most effective way to solve the problem.

Supporting Documentation:

AWS Documentation on Securing SageMaker Notebook Instances:
https://docs.aws.amazon.com/sagemaker/latest/dg/security-notebooks.html
AWS Identity and Access Management (IAM): https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html
Amazon VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html

Question: 142

Exam Heist

A company will use Amazon SageMaker to train and host a machine learning (ML) model for a marketing campaign. The majority of data is sensitive customer data. The data must be encrypted at rest. The company wants AWS to maintain the root of trust for the master keys and wants encryption key usage to be logged.

Which implementation will meet these requirements?

- Use encryption keys that are stored in AWS Cloud HSM to encrypt the ML data volumes, and to encrypt the model artifacts and data in Amazon S3.
- Use SageMaker built-in transient keys to encrypt the ML data volumes. Enable default encryption for new Amazon Elastic Block Store (Amazon EBS) volumes.
- Use customer managed keys in AWS Key Management Service (AWS KMS) to encrypt the ML data volumes, and to encrypt the model artifacts and data in Amazon S3. ✓
- Use AWS Security Token Service (AWS STS) to create temporary tokens to encrypt the ML storage volumes, and to encrypt the model artifacts and data in Amazon S3.

Explanation:

The correct answer is C because it aligns best with the stated requirements of encrypting sensitive customer data at rest, having AWS maintain the root of trust for the master keys, and logging encryption key usage.

Here's a detailed justification:

Encryption at rest: AWS KMS (Key Management Service) allows you to use customer-managed keys (CMKs) to encrypt data. These CMKs can be used to encrypt ML data volumes used by SageMaker training jobs and inference endpoints, and also to encrypt model artifacts and data stored in Amazon S3. This fulfills the requirement of encrypting data at rest.
AWS maintains root of trust: With customer-managed keys in KMS, AWS handles the creation, management, and protection of the CMKs within KMS. While the customer has control over the key policy and usage permissions, the underlying hardware security modules (HSMs) and infrastructure that protect the keys are managed by AWS. This means AWS maintains the root of trust.
Encryption key usage logging: KMS integrates with AWS CloudTrail. Every time a CMK is used for encryption or decryption, an entry is logged in CloudTrail. This allows the company to audit and track encryption key usage, fulfilling the logging requirement.
Let's analyze why the other options are not suitable:

A. Use encryption keys that are stored in AWS Cloud HSM: While CloudHSM provides enhanced control over the keys, including managing the HSMs themselves, it is more complex and costly than using KMS. Importantly, it does not align with the requirement that *AWS* maintain the root of trust. With CloudHSM, the customer is responsible for managing the HSM cluster.
B. Use SageMaker built-in transient keys: Transient keys are temporary and are not suitable for encrypting sensitive customer data intended to be protected at rest persistently. Enabling default encryption for EBS volumes might encrypt the OS volume, but wouldn't address the requirement of encrypting ML data volumes and S3 data.
D. Use AWS STS to create temporary tokens: AWS STS is used for granting temporary, limited-privilege access to AWS resources. It's not directly related to encrypting data at rest. While temporary tokens can be used in conjunction with encryption, they don't handle the key management and logging requirements specified in the question. STS is primarily for authentication and authorization, not encryption key management.
In summary, using customer-managed keys in AWS KMS provides the right balance of security, manageability, and logging, fulfilling all the requirements outlined in the question.
Supporting Documentation:

AWS Key Management Service (KMS): https://aws.amazon.com/kms/
Encryption at Rest with Amazon SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/encryption-at-rest.html
AWS CloudTrail: https://aws.amazon.com/cloudtrail/
AWS CloudHSM: https://aws.amazon.com/cloudhsm/
AWS Security Token Service (STS): https://aws.amazon.com/sts/

Question: 143

Exam Heist

A machine learning specialist stores IoT soil sensor data in Amazon DynamoDB table and stores weather event data as JSON files in Amazon S3. The dataset in
DynamoDB is 10 GB in size and the dataset in Amazon S3 is 5 GB in size. The specialist wants to train a model on this data to help predict soil moisture levels as a function of weather events using Amazon SageMaker.
Which solution will accomplish the necessary transformation to train the Amazon SageMaker model with the LEAST amount of administrative overhead?

- Launch an Amazon EMR cluster. Create an Apache Hive external table for the DynamoDB table and S3 data. Join the Hive tables and write the results out to Amazon S3.
- Crawl the data using AWS Glue crawlers. Write an AWS Glue ETL job that merges the two tables and writes the output to an Amazon Redshift cluster.
- Enable Amazon DynamoDB Streams on the sensor table. Write an AWS Lambda function that consumes the stream and appends the results to the existing weather files in Amazon S3.
- Crawl the data using AWS Glue crawlers. Write an AWS Glue ETL job that merges the two tables and writes the output in CSV format to Amazon S3. ✓

Explanation:
The correct answer is **D**. Here's why:
Rationale:
The core requirement is to combine data from DynamoDB and S3 to train a SageMaker model with minimal administrative overhead.

Option A (EMR with Hive): While EMR could handle this, it involves provisioning and managing an EMR cluster, which adds significant administrative overhead. Setting up and maintaining an EMR cluster is more complex than using a serverless ETL service like AWS Glue.
Option B (Glue to Redshift): Using Amazon Redshift introduces an additional managed service and adds to complexity, increasing administrative overhead. Redshift is primarily a data warehouse, and using it solely for data transformation for a

15GB dataset is an overkill.
Option C (DynamoDB Streams to Lambda): DynamoDB Streams with Lambda is suitable for real-time event processing, not for batch data transformation required for model training. It would be complex to correlate events from the stream with weather data retroactively, making it unsuitable for this scenario.
Option D (Glue ETL): AWS Glue is a fully managed ETL service. Glue crawlers can discover the schema of the DynamoDB table and S3 JSON data. The Glue ETL job can then perform the necessary transformations (joining, merging) and output the combined data in CSV format to S3. S3 is a suitable storage location for training data and is easily accessible by SageMaker. CSV is a common and efficient format for model training. This solution is serverless and requires the least amount of administrative overhead.
AWS Glue excels at this use case by offering schema discovery via crawlers, visual job authoring, and serverless execution of ETL jobs. It eliminates the need to manage underlying infrastructure.
Why other options are less suitable:

Higher Administrative Overhead: Options A and B involve managing compute clusters (EMR or Redshift).
Incorrect Use Case: Option C focuses on real-time processing, not the batch processing required for model training.
Efficiency and Simplicity: Glue ETL to S3 is generally the most streamlined and efficient solution for this type of data transformation.
Supporting Links:

AWS Glue Documentation: https://aws.amazon.com/glue/
Amazon SageMaker Documentation: https://aws.amazon.com/sagemaker/
AWS Glue ETL: https://docs.aws.amazon.com/glue/latest/dg/etl-tutorial-etl.html

Question: 144

Exam Heist

A company sells thousands of products on a public website and wants to automatically identify products with potential durability problems. The company has
1.000 reviews with date, star rating, review text, review summary, and customer email fields, but many reviews are incomplete and have empty fields. Each review has already been labeled with the correct durability result.
A machine learning specialist must train a model to identify reviews expressing concerns over product durability. The first model needs to be trained and ready to review in 2 days.
What is the MOST direct approach to solve this problem within 2 days?

- Train a custom classifier by using Amazon Comprehend. ✓
- Build a recurrent neural network (RNN) in Amazon SageMaker by using Gluon and Apache MXNet.
- Train a built-in BlazingText model using Word2Vec mode in Amazon SageMaker.
- Use a built-in seq2seq model in Amazon SageMaker.

Explanation:
The best approach to solve this problem within the tight 2-day deadline is to leverage Amazon Comprehend for custom classification. Here's why:
Amazon Comprehend is a fully managed natural language processing (NLP) service that allows you to build custom classification models without requiring deep machine learning expertise or significant coding. Option A aligns with the problem because custom classifiers within comprehend are able to analyze text data to identify patterns and categories, in this case identifying reviews that express concern over product durability.
Option B, building an RNN in SageMaker with Gluon and MXNet, is far too time-consuming. It involves writing a significant amount of code, tuning hyperparameters, and managing infrastructure, all of which cannot be accomplished in 2 days. While RNNs are powerful for sequence data like text, the development and deployment time is prohibitive.
Option C, training a BlazingText model with Word2Vec in SageMaker, also presents a challenge. Although BlazingText is faster than other algorithms, the time required to prepare the data, train the model, and deploy it exceeds the 2-day deadline. Additionally, this requires more ML expertise compared to using a service such as Comprehend.
Option D, using a seq2seq model in SageMaker, is primarily designed for sequence-to-sequence tasks, such as machine translation or text summarization, rather than text classification for sentiment analysis. This approach will also require more ML expertise and setup compared to the best approach using Comprehend.
Amazon Comprehend allows training a custom classification model using existing labeled data, which in this case, the labeled reviews are used for training. Comprehend manages the underlying infrastructure, reducing setup and management time. This allows quick development and deployment of a sentiment analysis model, fitting within the 2-day constraint.
Therefore, Option A is the most direct and fastest way to address the need to identify reviews expressing concerns over product durability, making it the best choice.

[Amazon Comprehend Custom Classification: https://docs.aws.amazon.com/comprehend/latest/dg/how-document-classification.html]

Question: 145

Exam Heist

A company that runs an online library is implementing a chatbot using Amazon Lex to provide book recommendations based on category. This intent is fulfilled by an AWS Lambda function that queries an Amazon DynamoDB table for a list of book titles, given a particular category. For testing, there are only three categories implemented as the custom slot types: "comedy," "adventure,` and "documentary.`
A machine learning (ML) specialist notices that sometimes the request cannot be fulfilled because Amazon Lex cannot understand the category spoken by users with utterances such as "funny," "fun," and "humor." The ML specialist needs to fix the problem without changing the Lambda code or data in DynamoDB.
How should the ML specialist fix the problem?

- Add the unrecognized words in the enumeration values list as new values in the slot type.
- Create a new custom slot type, add the unrecognized words to this slot type as enumeration values, and use this slot type for the slot.
- Use the AMAZON.SearchQuery built-in slot types for custom searches in the database.
- Add the unrecognized words as synonyms in the custom slot type. ✓

Explanation:
The best solution to the Amazon Lex chatbot issue is to add the unrecognized words as synonyms in the custom slot type. Here's why:

Understanding the Problem: The chatbot is failing to recognize user utterances like "funny," "fun," and "humor" as belonging to the "comedy" category. This is because Amazon Lex's slot filling mechanism isn't associating these words with the existing slot values ("comedy," "adventure," "documentary").
Why Synonyms are the Ideal Solution: Synonyms allow you to map different user utterances to the same slot value. By adding "funny," "fun," and "humor" as synonyms for "comedy" within the "comedy" slot type, Lex will understand that these words refer to the same category.
Why Other Options Are Less Suitable:

A. Adding as New Values: This is incorrect because "funny," "fun," and "humor" are not new categories, but rather different ways to express the "comedy" category. Adding them as new slot values would change the intended logic and potentially require modifications to the Lambda function and DynamoDB data.
B. Creating a New Slot Type: This is an overkill. The issue is not with the slot itself or the custom slot type, it is about lack of synonyms. The custom slot type is appropriate for the types of books needed.
C. Using AMAZON.SearchQuery: This slot type is for unconstrained text input, not for mapping to predefined categories. It would bypass the slot filling mechanism and require significant changes to the backend logic, making the Lambda function and DynamoDB look for words not in the table and is not desirable since we want to avoid those changes. The use of the AMAZON.SearchQuery slot type is more appropriate when users may be searching for details or items that may not be implemented in the slots, or the number of slot variables may be too extensive.
Benefits of Synonyms:

No Code Change: Synonyms are configured within the Amazon Lex console, requiring no changes to the AWS Lambda function or DynamoDB data.
Improved User Experience: The chatbot becomes more conversational and understanding, leading to a better user experience.
Maintainability: Synonyms are easy to manage and update, allowing you to adapt the chatbot to changing user language.
Cloud Computing Concepts:

Natural Language Understanding (NLU): Amazon Lex uses NLU to understand user intent and extract relevant information (slots).
Slot Filling: The process of identifying and extracting slot values from user utterances. Synonyms enhance the accuracy of slot filling.
Authoritative Links:

Amazon Lex Slots
Amazon Lex Slot Types

Question: 146

Exam Heist

A manufacturing company uses machine learning (ML) models to detect quality issues. The models use images that are taken of the company's product at the end of each production step. The company has thousands of machines at the production site that generate one image per second on average.
The company ran a successful pilot with a single manufacturing machine. For the pilot, ML specialists used an industrial PC that ran AWS IoT Greengrass with a long-running AWS Lambda function that uploaded the images to Amazon S3. The uploaded images invoked a Lambda function that was written in Python to perform inference by using an Amazon SageMaker endpoint that ran a custom model. The inference results were forwarded back to a web service that was hosted at the production site to prevent faulty products from being shipped.
The company scaled the solution out to all manufacturing machines by installing similarly configured industrial PCs on each production machine. However, latency for predictions increased beyond acceptable limits. Analysis shows that the internet connection is at its capacity limit.
How can the company resolve this issue MOST cost-effectively?

- Set up a 10 Gbps AWS Direct Connect connection between the production site and the nearest AWS Region. Use the Direct Connect connection to upload the images. Increase the size of the instances and the number of instances that are used by the SageMaker endpoint.
- Extend the long-running Lambda function that runs on AWS IoT Greengrass to compress the images and upload the compressed files to Amazon S3. Decompress the files by using a separate Lambda function that invokes the existing Lambda function to run the inference pipeline.
- Use auto scaling for SageMaker. Set up an AWS Direct Connect connection between the production site and the nearest AWS Region. Use the Direct Connect connection to upload the images.
- Deploy the Lambda function and the ML models onto the AWS IoT Greengrass core that is running on the industrial PCs that are installed on each machine. Extend the long-running Lambda function that runs on AWS IoT Greengrass to invoke the Lambda function with the captured images and run the inference on the edge component that forwards the results directly to the web service. ✓

Explanation:
The most cost-effective solution is **D: Deploy the Lambda function and the ML models onto the AWS IoT Greengrass core that is running on the industrial PCs that are installed on each machine. Extend the long-running Lambda function that runs on AWS IoT Greengrass to invoke the Lambda function with the captured images and run the inference on the edge component that forwards the results directly to the web service.**
Here's why:

Addressing the Bottleneck: The primary issue is the internet connection's capacity limit. Solution D avoids sending images to AWS for inference, which directly alleviates the network bottleneck.
Edge Computing: This approach leverages edge computing principles by performing inference locally on the industrial PCs using AWS IoT Greengrass. Edge computing reduces latency and dependence on network connectivity.
Cost Efficiency: This solution is cost-effective because it avoids the high costs associated with upgrading network infrastructure (Option A, C). It also avoids the complexity and potential cost increases with compressing and decompressing images (Option B). No additional infrastructure is needed because it utilizes the already deployed industrial PCs.
Greengrass Capabilities: AWS IoT Greengrass is designed for exactly this type of scenario. It allows you to deploy and manage Lambda functions and ML models on edge devices, providing local compute and inference capabilities.
Real-time Inference: By running inference locally, the solution minimizes latency, allowing for faster detection of quality issues and preventing faulty products from being shipped.
Reduced Cloud Costs: Eliminating image uploads to S3 and SageMaker inference reduces costs associated with storage, data transfer, and compute in the cloud.
Scalability and Maintainability: While initial deployment requires model deployment to each Greengrass core, AWS IoT Device Management and Greengrass group management simplify updates and model management across all devices.
Why other options are not the best:

A & C: Direct Connect addresses the bandwidth issue but is a significant expense. Increasing SageMaker instance size also increases costs when edge inference is a better alternative.
B: Compression might reduce bandwidth usage, but adds complexity in managing compression/decompression and doesn't fundamentally eliminate the need to transfer images.
Supporting Links:

AWS IoT Greengrass: https://aws.amazon.com/greengrass/
Edge Computing: https://aws.amazon.com/edge-computing/

Question: 147

Exam Heist

A data scientist is using an Amazon SageMaker notebook instance and needs to securely access data stored in a specific Amazon S3 bucket.
How should the data scientist accomplish this?

- Add an S3 bucket policy allowing GetObject, PutObject, and ListBucket permissions to the Amazon SageMaker notebook ARN as principal.
- Encrypt the objects in the S3 bucket with a custom AWS Key Management Service (AWS KMS) key that only the notebook owner has access to.

- Attach the policy to the IAM role associated with the notebook that allows GetObject, PutObject, and ListBucket operations to the specific S3 bucket. ✓
- Use a script in a lifecycle configuration to configure the AWS CLI on the instance with an access key ID and secret.

Explanation:

The correct answer is C: Attach the policy to the IAM role associated with the notebook that allows GetObject, PutObject, and ListBucket operations to the specific S3 bucket.

Here's a detailed justification:

IAM roles are the standard and secure way to grant permissions to AWS resources, like a SageMaker notebook instance. When a SageMaker notebook instance is launched, it's associated with an IAM role. This role provides temporary credentials that the notebook can use to access other AWS services. By attaching an IAM policy to this role, we directly grant the notebook instance the necessary permissions to access the S3 bucket. This approach follows the principle of least privilege by granting only the required permissions (GetObject, PutObject, ListBucket) to a specific bucket, reducing the risk of unauthorized access to other resources. Using IAM roles is a best practice for managing access control within AWS.

Option A is incorrect because modifying S3 bucket policies to grant access to individual ARNs, although functional, is less manageable and scalable compared to using IAM roles associated with the notebook instance. It's better to control access through the IAM role assigned to the compute resource than by granting explicit permissions in bucket policies.

Option B is incorrect because while encrypting the S3 objects with KMS enhances security, it doesn't address the fundamental need to grant the notebook instance permission to access the S3 bucket. Even with encryption, the notebook needs IAM permissions to decrypt and access the data. Furthermore, restricting access to a custom KMS key to only the notebook owner might hinder collaboration or automated processes that need access.

Option D is incorrect as it involves manually configuring the AWS CLI with access key ID and secret, a highly discouraged practice. Storing access keys directly on an instance is a security risk and should be avoided. IAM roles provide a much more secure and manageable way to grant temporary credentials without needing to manage long-term secrets. This method also doesn't automatically rotate the credentials, increasing the risk of compromised credentials. IAM roles provide automatically rotated, short-term credentials.

In summary, using IAM roles to grant permissions to a SageMaker notebook instance for S3 access is the most secure, manageable, and scalable approach. It adheres to best practices for AWS security and simplifies access control.

Further research:

IAM Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
SageMaker Roles: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html
IAM Best Practices: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Question: 148

Exam Heist

A company is launching a new product and needs to build a mechanism to monitor comments about the company and its new product on social media. The company needs to be able to evaluate the sentiment expressed in social media posts, and visualize trends and configure alarms based on various thresholds.

The company needs to implement this solution quickly, and wants to minimize the infrastructure and data science resources needed to evaluate the messages.

The company already has a solution in place to collect posts and store them within an Amazon S3 bucket.

What services should the data science team use to deliver this solution?

- Train a model in Amazon SageMaker by using the BlazingText algorithm to detect sentiment in the corpus of social media posts. Expose an endpoint that can be called by AWS Lambda. Trigger a Lambda function when posts are added to the S3 bucket to invoke the endpoint and record the sentiment in an Amazon DynamoDB table and in a custom Amazon CloudWatch metric. Use CloudWatch alarms to notify analysts of trends.
- Train a model in Amazon SageMaker by using the semantic segmentation algorithm to model the semantic content in the corpus of social media posts. Expose an endpoint that can be called by AWS Lambda. Trigger a Lambda function when objects are added to the S3 bucket to invoke the endpoint and record the sentiment in an Amazon DynamoDB table. Schedule a second Lambda function to query recently added records and send an Amazon Simple Notification Service (Amazon SNS) notification to notify analysts of trends.
- Trigger an AWS Lambda function when social media posts are added to the S3 bucket. Call Amazon Comprehend for each post to capture the sentiment in themessage and record the sentiment in an Amazon DynamoDB table. Schedule a second Lambda function to query recently added records and send an Amazon Simple Notification Service (Amazon SNS) notification to notify analysts of trends.
- Trigger an AWS Lambda function when social media posts are added to the S3 bucket. Call Amazon Comprehend for each post to capture the sentiment in the message and record the sentiment in a custom Amazon CloudWatch metric and in S3. Use CloudWatch alarms to notify analysts of trends. ✓

Explanation:

The best solution utilizes Amazon Comprehend to quickly analyze sentiment and leverages CloudWatch for monitoring and alarming, minimizing infrastructure and data science effort.
Here's why option D is the correct choice:

Minimize Infrastructure and Data Science Effort: Comprehend is a pre-trained NLP service, removing the need to train and deploy a custom sentiment analysis model. This fulfills the requirement to minimize both infrastructure and data science resources. https://aws.amazon.com/comprehend/
Real-time Sentiment Analysis: An AWS Lambda function triggered by S3 events allows for real-time processing of newly added social media posts. This aligns with the requirement of monitoring comments as they are posted.
Sentiment Analysis: Comprehend accurately identifies the sentiment (positive, negative, neutral, mixed) within the social media posts.
Monitoring and Alarming: Storing sentiment data as a custom CloudWatch metric provides a way to visualize trends and set alarms based on specified thresholds. This directly addresses the monitoring and alarming requirement.
Cost-Effectiveness: Using Comprehend is typically more cost-effective than building and maintaining a custom model, especially for initial deployments and fluctuating workloads.
Here's why the other options are less suitable:

Option A & B: Training a custom model with SageMaker (BlazingText or Semantic Segmentation) requires significant data science effort, model training time, and infrastructure maintenance. Semantic Segmentation is not suitable for sentiment analysis. https://aws.amazon.com/sagemaker/
Option C: While it uses Comprehend, it lacks direct integration with CloudWatch metrics for visualization and alerting. The need to schedule a Lambda function and send SNS notifications adds unnecessary complexity. Also storing the result in S3 is useless. Cloudwatch Metrics provides a native solution for the use case.
Option D efficiently uses serverless services and a pre-trained NLP model to meet all requirements with minimal effort and infrastructure.

Question: 149

Exam Heist

A bank wants to launch a low-rate credit promotion. The bank is located in a town that recently experienced economic hardship. Only some of the bank's customers were affected by the crisis, so the bank's credit team must identify which customers to target with the promotion. However, the credit team wants to make sure that loyal customers' full credit history is considered when the decision is made.
The bank's data science team developed a model that classifies account transactions and understands credit eligibility. The data science team used the XGBoost algorithm to train the model. The team used 7 years of bank transaction historical data for training and hyperparameter tuning over the course of several days.
The accuracy of the model is sufficient, but the credit team is struggling to explain accurately why the model denies credit to some customers. The credit team has almost no skill in data science.
What should the data science team do to address this issue in the MOST operationally efficient manner?

- Use Amazon SageMaker Studio to rebuild the model. Create a notebook that uses the XGBoost training container to perform model training. Deploy the model at an endpoint. Enable Amazon SageMaker Model Monitor to store inferences. Use the inferences to create Shapley values that help explain model behavior. Create a chart that shows features and SHapley Additive exPlanations (SHAP) values to explain to the credit team how the features affect the model outcomes.
- Use Amazon SageMaker Studio to rebuild the model. Create a notebook that uses the XGBoost training container to perform model training. Activate Amazon SageMaker Debugger, and configure it to calculate and collect Shapley values. Create a chart that shows features and SHapley Additive exPlanations (SHAP) values to explain to the credit team how the features affect the model outcomes. ✓
- Create an Amazon SageMaker notebook instance. Use the notebook instance and the XGBoost library to locally retrain the model. Use the plot_importance() method in the Python XGBoost interface to create a feature importance chart. Use that chart to explain to the credit team how the features affect the model outcomes.
- Use Amazon SageMaker Studio to rebuild the model. Create a notebook that uses the XGBoost training container to perform model training. Deploy the model at an endpoint. Use Amazon SageMaker Processing to post-analyze the model and create a feature importance explainability chart automatically for the credit team.

Explanation:
The correct answer is **B**. Here's why:
The core problem is the lack of explainability of the XGBoost model's predictions to a non-technical credit team. The best solution will provide clear, understandable explanations in an operationally efficient way. Shapley values offer a strong method for explaining individual predictions.
Why B is the best option:

SageMaker Studio: Provides a collaborative environment for model building and explanation.
XGBoost Training Container: Leverages a pre-built, optimized environment for XGBoost, reducing setup overhead.
SageMaker Debugger with Shapley Value Calculation: This is crucial. SageMaker Debugger is designed to profile and debug training jobs. A key feature is its ability to compute and collect SHAP (SHapley Additive exPlanations) values directly during

training or after the model has been trained. This allows for a detailed understanding of feature importance for each prediction.
SHAP Chart for Explanation: Visualizing SHAP values in a chart provides a readily understandable way for the credit team to see how different features affect the model's output (credit approval/denial). SHAP values explain how each feature contributes to pushing the prediction away from the baseline or average prediction.
Operational Efficiency: Using SageMaker Debugger for Shapley calculation is more efficient than post-hoc calculation (option A) or local retraining (option C) because it directly integrates with the training process and can be optimized for the distributed environment.
Why other options are less ideal:

A: Using SageMaker Model Monitor for inferences and then creating Shapley values adds unnecessary steps and complexity. Model Monitor focuses on detecting model drift and data quality issues after deployment, not primarily on model explainability. Also, Model Monitor would require a separate step of calculating the Shapley values *after* inference, which is less operationally efficient than having Debugger compute them during training.
C: Retraining the model locally on a notebook instance using the `plot_importance()` method is insufficient. `plot_importance()` provides *global* feature importance, not explanation for individual predictions. Also, it involves setting up a local environment which reduces operational efficiency and scalability. The feature importance from XGBoost doesn't provide individual explanation.
D: SageMaker Processing for post-analysis to create a feature importance chart lacks the granularity of SHAP values. Processing jobs are more suited for data preparation or feature engineering and may not be the optimal tool for sophisticated explainability analysis, also it's less clear on how this would happen "automatically" as it's described in the option.
Key Concepts and Links:

SHAP Values: A unified measure of feature importance, explaining how each feature contributes to the prediction. https://github.com/slundberg/shap
Amazon SageMaker Debugger: A tool for profiling and debugging machine learning models. https://aws.amazon.com/sagemaker/debugger/
Amazon SageMaker Studio: An integrated development environment (IDE) for machine learning. https://aws.amazon.com/sagemaker/studio/
XGBoost: A popular gradient boosting algorithm. https://xgboost.readthedocs.io/en/stable/
In summary, option B provides the most effective and operationally efficient solution for explaining the XGBoost model's decisions to a non-technical audience by leveraging SageMaker Debugger to calculate and visualize SHAP values.

Question: 150

Exam Heist

A data science team is planning to build a natural language processing (NLP) application. The application's text preprocessing stage will include part-of-speech tagging and key phase extraction. The preprocessed text will be input to a custom classification algorithm that the data science team has already written and trained using Apache MXNet.
Which solution can the team build MOST quickly to meet these requirements?

- Use Amazon Comprehend for the part-of-speech tagging, key phase extraction, and classification tasks.
- Use an NLP library in Amazon SageMaker for the part-of-speech tagging. Use Amazon Comprehend for the key phase extraction. Use AWS Deep Learning Containers with Amazon SageMaker to build the custom classifier.
- Use Amazon Comprehend for the part-of-speech tagging and key phase extraction tasks. Use Amazon SageMaker built-in Latent Dirichlet Allocation (LDA) algorithm to build the custom classifier.
- Use Amazon Comprehend for the part-of-speech tagging and key phase extraction tasks. Use AWS Deep Learning Containers with Amazon SageMaker to build the custom classifier. ✓

Explanation:
The question asks for the *fastest* way to build an NLP application involving part-of-speech tagging, key phrase extraction, and a custom MXNet-based classifier.
Option D is the most efficient because it leverages Amazon Comprehend for the NLP preprocessing tasks (part-of-speech tagging and key phrase extraction). Comprehend is a fully managed NLP service, allowing quick implementation without infrastructure management. For the custom classifier, AWS Deep Learning Containers within SageMaker enables the deployment of pre-trained MXNet models. This avoids the need to retrain or re-implement the existing model, saving significant time.
Option A suggests using Comprehend for classification, but the team already has a trained MXNet model. Replacing it with Comprehend's built-in capabilities would require retraining and validation, impacting speed.
Option B is less efficient as it distributes NLP tasks between a library in SageMaker and Comprehend. It also still requires using Deep Learning Containers to deploy the custom MXNet model, making it less efficient than D overall as well.

Option C suggests using Comprehend for preprocessing and SageMaker's LDA for classification. LDA is an unsupervised topic modeling algorithm, unsuitable for a pre-trained custom *classification* model. Moreover, converting and reimplementing the custom classification algorithm to utilize LDA would be time-consuming, violating the prompt's emphasis on speed.
Therefore, Option D is the best solution because it combines the speed of managed services (Comprehend) with the flexibility of deploying custom models (AWS Deep Learning Containers) without requiring retraining or reimplementation. It capitalizes on the existing investment in the custom MXNet classifier.
Relevant Links:

Amazon Comprehend: https://aws.amazon.com/comprehend/
AWS Deep Learning Containers: https://aws.amazon.com/revisions/aws-deep-learning-containers/
Amazon SageMaker: https://aws.amazon.com/sagemaker/

Question: 151

Exam Heist

A machine learning (ML) specialist must develop a classification model for a financial services company. A domain expert provides the dataset, which is tabular with 10,000 rows and 1,020 features. During exploratory data analysis, the specialist finds no missing values and a small percentage of duplicate rows. There are correlation scores of > 0.9 for 200 feature pairs. The mean value of each feature is similar to its 50th percentile.
Which feature engineering strategy should the ML specialist use with Amazon SageMaker?

- ◯ Apply dimensionality reduction by using the principal component analysis (PCA) algorithm. ✓
- ◯ Drop the features with low correlation scores by using a Jupyter notebook.
- ◯ Apply anomaly detection by using the Random Cut Forest (RCF) algorithm.
- ◯ Concatenate the features with high correlation scores by using a Jupyter notebook.

Explanation:
Here's a detailed justification for choosing Principal Component Analysis (PCA) in this scenario:
The problem describes a dataset with a high number of features (1020) and significant multicollinearity (200 feature pairs with correlation > 0.9). This situation, known as the "curse of dimensionality," can lead to overfitting, increased computational cost, and difficulty in interpreting the model. The fact that the mean and median are similar suggests a near-normal distribution of the features, which benefits PCA.
PCA is a dimensionality reduction technique that aims to transform the original features into a new set of uncorrelated features called principal components. These components are ordered by the amount of variance they explain in the data. By selecting a subset of the top principal components, the ML specialist can significantly reduce the dimensionality of the dataset while preserving most of the important information. This simplification addresses the multicollinearity issue and reduces the risk of overfitting.
Option B is incorrect because dropping features based solely on "low correlation scores" without considering their individual predictive power is a poor strategy. Features with low correlation *to each other* might still be very predictive of the target variable. Option C, anomaly detection using Random Cut Forest (RCF), is not relevant to the core problem of dimensionality reduction and multicollinearity. RCF is used for identifying outliers, not for feature engineering in this specific scenario. Option D, concatenating highly correlated features, would exacerbate multicollinearity issues, making the model more unstable and difficult to interpret.
SageMaker provides built-in algorithms and libraries that support PCA. The Scikit-learn library, readily available within SageMaker notebooks and training environments, offers a straightforward implementation of PCA. Using PCA within SageMaker allows the ML specialist to preprocess the data and train a more robust and efficient model. This is more efficient than hand-coding feature concatenation or complex feature selection within a Jupyter Notebook.
In conclusion, PCA is the most appropriate feature engineering strategy because it directly addresses the issues of high dimensionality and multicollinearity in a statistically sound manner, leading to a simpler, more robust, and interpretable model.
Further Research:

Amazon SageMaker PCA: https://docs.aws.amazon.com/sagemaker/latest/dg/pca_HowItWorks.html
Scikit-learn PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Curse of Dimensionality: https://en.wikipedia.org/wiki/Curse_of_dimensionality

Question: 152

Exam Heist

A manufacturing company asks its machine learning specialist to develop a model that classifies defective parts into one of eight defect types. The company has provided roughly 100,000 images per defect type for training. During the initial training of the image classification model, the specialist notices that the validation accuracy is 80%, while the training accuracy is 90%. It is known that human-level performance for this type of

image classification is around 90%.
What should the specialist consider to fix this issue?

- A longer training time
- Making the network larger
- Using a different optimizer
- Using some form of regularization ✓

Explanation:
The problem described exhibits overfitting. The model performs well on the training data (90% accuracy) but worse on the validation data (80% accuracy). This indicates that the model has learned the training data too well, including its noise and specific nuances, and struggles to generalize to unseen data.
Option D, "Using some form of regularization," directly addresses overfitting. Regularization techniques, such as L1 or L2 regularization, add a penalty term to the loss function, discouraging the model from learning overly complex patterns. This prevents the model from memorizing the training data and encourages it to learn more generalizable features, improving performance on the validation set. Dropout is another form of regularization that randomly deactivates neurons during training, further preventing overfitting.
Options A and B are less likely to address the root cause. A longer training time (Option A) without addressing the overfitting problem could worsen the situation, leading to even lower validation accuracy. Making the network larger (Option B) could also exacerbate overfitting by increasing the model's capacity to memorize the training data. Option C, using a different optimizer, might lead to faster or more stable training, but it doesn't directly combat overfitting; while some optimizers might implicitly provide some regularization, it's not their primary function.
Therefore, regularization is the most appropriate method to address the described overfitting issue and improve the generalization performance of the image classification model. By penalizing complexity, regularization encourages the model to learn more robust and generalizable features, thereby narrowing the gap between training and validation accuracy.
For further research, consider exploring these resources:

Overfitting and Underfitting: https://www.ibm.com/cloud/learn/overfitting-underfitting
Regularization: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/introduction
Dropout Regularization: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

Question: 153

Exam Heist

A machine learning specialist needs to analyze comments on a news website with users across the globe. The specialist must find the most discussed topics in the comments that are in either English or Spanish.
What steps could be used to accomplish this task? (Choose two.)

- Use an Amazon SageMaker BlazingText algorithm to find the topics independently from language. Proceed with the analysis.
- Use an Amazon SageMaker seq2seq algorithm to translate from Spanish to English, if necessary. Use a SageMaker Latent Dirichlet Allocation (LDA) algorithm to find the topics.
- Use Amazon Translate to translate from Spanish to English, if necessary. Use Amazon Comprehend topic modeling to find the topics. ✓
- Use Amazon Translate to translate from Spanish to English, if necessary. Use Amazon Lex to extract topics form the content.
- Use Amazon Translate to translate from Spanish to English, if necessary. Use Amazon SageMaker Neural Topic Model (NTM) to find the topics.

Explanation:
The correct answers are B and C. Here's why:

Why C is correct:

Amazon Translate: This service is designed for accurate and scalable language translation. Translating all Spanish comments to English allows for a unified analysis of the entire dataset, regardless of the original language. This is crucial for finding topics that span both language groups. https://aws.amazon.com/translate/
Amazon Comprehend: Comprehend provides natural language processing (NLP) capabilities, including topic modeling. It can automatically identify frequently discussed topics within a collection of documents. This eliminates the need to build and train a custom topic modeling solution. https://aws.amazon.com/comprehend/
Combining these allows for translation and immediate topic extraction using a built-in service.
Why B is also correct:

Amazon SageMaker seq2seq: A sequence-to-sequence model can handle the translation from Spanish to English.

SageMaker Latent Dirichlet Allocation (LDA): LDA is a topic modeling algorithm that identifies clusters of words (topics) from a collection of documents. After translation by seq2seq, LDA can be used to identify the most discussed topics.
Why the other options are incorrect:

A: BlazingText is useful for word embeddings and text classification but not for topic discovery independently of language. Languages have different words and vocabulary.
D: Amazon Lex is primarily for building conversational interfaces (chatbots), not for large-scale topic extraction from text.
E: While Amazon SageMaker Neural Topic Model (NTM) can perform topic modeling, using Comprehend is more straightforward for the task when Translate is also used, as they are related services.

Question: 154

Exam Heist

A machine learning (ML) specialist is administering a production Amazon SageMaker endpoint with model monitoring configured. Amazon SageMaker Model
Monitor detects violations on the SageMaker endpoint, so the ML specialist retrains the model with the latest dataset. This dataset is statistically representative of the current production traffic. The ML specialist notices that even after deploying the new SageMaker model and running the first monitoring job, the SageMaker endpoint still has violations.
What should the ML specialist do to resolve the violations?

- Manually trigger the monitoring job to re-evaluate the SageMaker endpoint traffic sample.
- Run the Model Monitor baseline job again on the new training set. Configure Model Monitor to use the new baseline. ✓
- Delete the endpoint and recreate it with the original configuration.
- Retrain the model again by using a combination of the original training set and the new training set.

Explanation:
The correct answer is B: Run the Model Monitor baseline job again on the new training set. Configure Model Monitor to use the new baseline.
Here's why:
Model Monitor detects violations by comparing the characteristics of the incoming data to a baseline profile that represents the expected behavior of the model. If the model has been retrained, the previous baseline may no longer be appropriate. The new model learns new data distribution. The incoming production traffic may now align with the characteristics of the new model's training data, but if the monitoring is still evaluated against the old baseline, it will continue to trigger false positive violations.
Option A is incorrect because manually triggering the same monitoring job won't help if the baseline is outdated. The job will still use the old baseline and, therefore, continue to report violations.
Option C is incorrect because recreating the endpoint with the original configuration won't address the underlying issue of an outdated baseline. The problem lies in the mismatch between the model's expected data distribution (as represented by the old baseline) and the actual distribution of the current data (as reflected in the new training dataset). The new model learns new data distribution.
Option D is less ideal because while retraining the model using a combination of the original and new datasets might improve performance, it doesn't directly address the issue of the incorrect baseline used for monitoring. The core problem is that the baseline must be updated to reflect the distribution of the new data and model. Furthermore, it is not necessary to retrain again before updating the baseline.
Therefore, the correct approach is to rerun the baseline job on the new training dataset to create a new baseline profile that accurately represents the expected behavior of the retrained model. After creating the new baseline, Model Monitor must be configured to use it for future monitoring jobs. This ensures that violations are detected only when there's a true deviation from the expected behavior of the retrained model.
Amazon SageMaker Model Monitor documentationCreating Baselines for Monitoring Data

Question: 155

Exam Heist

A company supplies wholesale clothing to thousands of retail stores. A data scientist must create a model that predicts the daily sales volume for each item for each store. The data scientist discovers that more than half of the stores have been in business for less than 6 months. Sales data is highly consistent from week to week. Daily data from the database has been aggregated weekly, and weeks with no sales are omitted from the current dataset. Five years (100 MB) of sales data is available in Amazon S3.
Which factors will adversely impact the performance of the forecast model to be developed, and which actions should the data scientist take to mitigate them?
(Choose two.)

- [x] Detecting seasonality for the majority of stores will be an issue. Request categorical data to relate new stores with similar stores that have more historical data. ✓
- [ ] The sales data does not have enough variance. Request external sales data from other industries to improve the model's ability to generalize.
- [x] Sales data is aggregated by week. Request daily sales data from the source database to enable building a daily model. ✓
- [ ] The sales data is missing zero entries for item sales. Request that item sales data from the source database include zero entries to enable building the model.
- [ ] Only 100 MB of sales data is available in Amazon S3. Request 10 years of sales data, which would provide 200 MB of training data for the model.

Explanation:
Here's a detailed justification for why options A and C are the best choices, and why the others aren't, within the constraints of your request:
Why A is correct:

Seasonality Detection Difficulty: Many stores having less than 6 months of data (less than two full seasons) makes it challenging to accurately identify and model seasonal patterns in their sales. Seasonality is a crucial factor for forecasting clothing sales.
Mitigation with Categorical Data: Requesting categorical data (e.g., store demographics, location, size, target customer segment) allows the data scientist to group newer stores with older, similar stores. By leveraging the historical data of analogous stores, the model can better estimate seasonality for the newer stores. This addresses the lack of historical data for new stores.
Why C is correct:

Aggregation Loss: Aggregating sales data weekly reduces the granularity and masks potentially important daily fluctuations. Daily sales might be influenced by promotions, events, or other factors that are lost in weekly aggregates.
Daily Model Requirement: The problem explicitly requires predicting *daily* sales volume. Weekly aggregated data makes this impossible without disaggregation (which is often inaccurate). Obtaining daily data from the source enables building the required daily model.
Why B is incorrect:

Unnecessary External Data: The problem doesn't suggest a lack of diversity in clothing styles or store types. Injecting data from "other industries" is likely irrelevant and could introduce noise, making it harder to find relevant patterns in the clothing sales data. The focus should be on improving the quality and granularity of *existing* sales data.
Why D is incorrect:

Implicit Zero Sales: Omitting weeks with no sales does not mean they need to be explicitly requested. Zero sales are easily represented by introducing zero values in the current dataset for those item sales, or by leveraging time series techniques that can handle missing data.
Why E is incorrect:

Data Volume Sufficient (Potentially): 100MB of five years of weekly sales data is not necessarily insufficient. While more data is usually helpful, increasing the dataset to 200MB might not be the primary issue or most impactful improvement. The focus should be on the quality and granularity of the existing data first. Simply doubling the amount of the same type of data might not solve the underlying problems of seasonality detection and daily prediction.
In Summary:
The most pressing issues are the difficulty in detecting seasonality for new stores and the need for daily data to fulfill the prediction requirement. Addressing these issues by gathering relevant categorical data and obtaining daily sales data from the database are the most appropriate actions.
Relevant Resources:

Time Series Analysis: https://otexts.com/fpp3/
Amazon Forecast: https://aws.amazon.com/forecast/
Feature Engineering: https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

Question: 156

Exam Heist

An ecommerce company is automating the categorization of its products based on images. A data scientist has trained a computer vision model using the Amazon
SageMaker image classification algorithm. The images for each product are classified according to specific product lines. The accuracy of the model is too low when categorizing new products. All of the product images have the same dimensions and are stored within an Amazon S3 bucket.

The company wants to improve the model so it can be used for new products as soon as possible.
Which steps would improve the accuracy of the solution? (Choose three.)

- [ ] Use the SageMaker semantic segmentation algorithm to train a new model to achieve improved accuracy.
- [ ] Use the Amazon Rekognition DetectLabels API to classify the products in the dataset.
- [x] Augment the images in the dataset. Use open source libraries to crop, resize, flip, rotate, and adjust the brightness and contrast of the images. ✓
- [ ] Use a SageMaker notebook to implement the normalization of pixels and scaling of the images. Store the new dataset in Amazon S3.
- [x] Use Amazon Rekognition Custom Labels to train a new model. ✓
- [x] Check whether there are class imbalances in the product categories, and apply oversampling or undersampling as required. Store the new dataset in Amazon S3. ✓

Explanation:
The correct answer is CEF because these options directly address the identified problem of low model accuracy when categorizing new products.
C. Augment the images in the dataset: Data augmentation is a crucial technique for improving the generalization ability of computer vision models. By applying transformations like cropping, resizing, flipping, and adjusting brightness and contrast, we can artificially increase the size and diversity of the training dataset. This helps the model become more robust to variations in image quality, lighting conditions, and object orientation, leading to improved performance on unseen data (new products). https://www.tensorflow.org/tutorials/images/data_augmentation
E. Use Amazon Rekognition Custom Labels to train a new model: Rekognition Custom Labels is a managed service specifically designed for training custom image classification models. It simplifies the process of building and deploying custom models without requiring extensive machine learning expertise. It can be a viable alternative if the existing SageMaker image classification algorithm isn't providing sufficient accuracy, as Rekognition can automatically optimize the model for the given dataset. https://aws.amazon.com/rekognition/custom-labels-features/
F. Check whether there are class imbalances in the product categories, and apply oversampling or undersampling as required. Store the new dataset in Amazon S3: Class imbalance, where some product categories have significantly more images than others, can negatively impact model performance. The model may be biased towards the majority classes and perform poorly on the minority classes (less represented product lines). By identifying and addressing class imbalances using techniques like oversampling (increasing the number of samples in minority classes) or undersampling (reducing the number of samples in majority classes), we can ensure the model learns more effectively from all product categories.
https://developers.google.com/machine-learning/data-validation/solution/imbalanced-data
Why other options are incorrect:

A. Use the SageMaker semantic segmentation algorithm to train a new model to achieve improved accuracy. Semantic segmentation is primarily used to classify each pixel in an image, which isn't directly relevant to the task of categorizing the entire product image.
B. Use the Amazon Rekognition DetectLabels API to classify the products in the dataset. Rekognition DetectLabels provides general-purpose object and scene detection, which is not suitable for classifying product images into specific product lines, because it will provide generic labels and not specific product lines.
D. Use a SageMaker notebook to implement the normalization of pixels and scaling of the images. Store the new dataset in Amazon S3. While normalization and scaling are essential preprocessing steps, the problem statement doesn't indicate that pixel ranges or image sizes are the primary cause of the accuracy issues. Therefore, it is not as vital as the other options (C, E, and F) that tackle data quality and algorithm suitability issues.

Question: 157

Exam Heist

A data scientist is training a text classification model by using the Amazon SageMaker built-in BlazingText algorithm. There are 5 classes in the dataset, with 300 samples for category A, 292 samples for category B, 240 samples for category C, 258 samples for category D, and 310 samples for category E.
The data scientist shuffles the data and splits off 10% for testing. After training the model, the data scientist generates confusion matrices for the training and test sets.

Training data confusion matrix

		Predicted class					
		A	B	C	D	E	Total
True class	A	270	0	0	0	0	270
	B	1	260	0	0	2	263
	C	0	0	111	100	5	216
	D	4	3	132	92	1	232
	E	0	0	2	3	274	279
	Total	275	263	245	195	282	1260

Test data confusion matrix

		Predicted class					
		A	B	C	D	E	Total
True class	A	9	1	0	0	0	10
	B	2	25	0	2	0	29
	C	10	2	11	10	1	34
	D	1	0	12	14	0	27
	E	9	1	4	1	25	40
	Total	31	29	27	27	26	140

What could the data scientist conclude form these results?

- Classes C and D are too similar. ✓
- The dataset is too small for holdout cross-validation.
- The data distribution is skewed.
- The model is overfitting for classes B and E.

Explanation:

The correct answer be A, the model is clearly unable to tell C and D apartthe reason why B is incorrect is subtle - there is holdout validation or cross-validation, but not holdout cross-validation; while I think it would be more reasonable to use CV with such a small dataset rather than holdout, the answer is mixing terms and therefore should be wrongalso, the test set confusion matrix is still pretty comparable to the train set one, so I wouldn't say there is objective evidence to claim holdout is a wrong choice here.

Question: 158 Exam Heist

A company that manufactures mobile devices wants to determine and calibrate the appropriate sales price for its devices. The company is collecting the relevant data and is determining data features that it can use to train machine learning (ML) models. There are more than 1,000 features, and the company wants to determine the primary features that contribute to the sales price.
Which techniques should the company use for feature selection? (Choose three.)

- [ ] Data scaling with standardization and normalization
- [x] Correlation plot with heat maps ✓
- [ ] Data binning
- [x] Univariate selection ✓
- [x] Feature importance with a tree-based classifier ✓
- [ ] Data augmentation

Explanation:

Here's a detailed justification for why options B, D, and E are the most appropriate feature selection techniques for the given scenario:

B. Correlation plot with heat maps: Correlation plots using heatmaps visually represent the relationships between different features. This allows the company to identify highly correlated features and potentially remove redundant ones. High multicollinearity (high correlation between independent variables) can negatively impact model performance. By visualizing these correlations, the company can make informed decisions about which features to keep, prioritizing features that have a strong correlation with the target variable (sales price) but low correlation with each other. This simplifies the model and can improve interpretability.

D. Univariate selection: Univariate feature selection involves evaluating each feature individually against the target variable using statistical tests (e.g., chi-squared test for categorical features, ANOVA for numerical features). These tests provide a measure of how well each feature predicts the target variable. Features with low scores or p-values above a certain threshold can be eliminated. This method is simple and computationally efficient, making it suitable for datasets with a large number of features. Scikit-learn in Python offers tools like `SelectKBest` which implement various univariate selection methods.

E. Feature importance with a tree-based classifier: Tree-based models like Random Forests or Gradient Boosted Trees inherently provide a measure of feature importance. During the training process, these models evaluate which features are most effective at splitting the data and reducing impurity (e.g., variance or entropy). The resulting feature importance scores indicate the relative contribution of each feature to the model's predictive power. Features with low importance scores can be considered for removal. This method is particularly useful because it captures non-linear relationships between features and the target variable, which linear methods might miss.

Why other options are less suitable:

A. Data scaling with standardization and normalization: While data scaling is essential for many ML algorithms, it doesn't inherently perform feature selection. Scaling adjusts the range of feature values but doesn't eliminate features.

C. Data binning: Data binning (discretization) transforms continuous variables into categorical ones. It's a data preprocessing technique, not a feature selection method. While binning can sometimes reveal non-linear relationships, it doesn't directly help in reducing the number of features.

F. Data augmentation: Data augmentation involves creating new synthetic data points based on existing ones (e.g., by adding noise, rotating images). It aims to improve model robustness and generalization, but it doesn't reduce the number of features.

Authoritative Links:

Scikit-learn Feature Selection: https://scikit-learn.org/stable/modules/feature_selection.html
Correlation and Heatmaps (Seaborn): https://seaborn.pydata.org/examples/many_pairwise_correlations.html
Feature Importance with Random Forests: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Therefore, the combination of correlation plots (B), univariate selection (D), and feature importance from tree-based classifiers (E) provides a comprehensive approach to identify and select the most relevant features for predicting the sales price of mobile devices from a dataset of 1,000+ features.

Question: 159

Exam Heist

A power company wants to forecast future energy consumption for its customers in residential properties and commercial business properties. Historical power consumption data for the last 10 years is available. A team of data scientists who performed the initial data analysis and feature selection will include the historical power consumption data and data such as weather, number of individuals on the property, and public holidays. The data scientists are using Amazon Forecast to generate the forecasts.
Which algorithm in Forecast should the data scientists use to meet these requirements?

- Autoregressive Integrated Moving Average (AIRMA)
- Exponential Smoothing (ETS)
- Convolutional Neural Network - Quantile Regression (CNN-QR) ✓
- Prophet

Explanation:

The correct answer is C, Convolutional Neural Network - Quantile Regression (CNN-QR). Here's why:
CNN-QR excels in scenarios with complex, non-linear relationships within time series data, which is highly probable when forecasting energy consumption. The historical data includes factors like weather, number of individuals, and holidays. These factors are likely to interact in complex and non-linear ways to influence energy consumption. CNN-QR can capture these intricate patterns because of its ability to learn hierarchical representations from the input data using convolutional layers.

Also, quantile regression estimates conditional quantiles of the response variable, which provides a range of possible outcomes, not just a single point forecast, thus supplying valuable information about uncertainty. The power company can use the range of possible energy demands to plan for different scenarios.
ARIMA (A) and ETS (B) are classical time series models. They are suitable when there's no additional data available beyond the time series data. They may struggle with the complexity introduced by the additional features of weather, number of individuals and holidays because they focus primarily on autocorrelations and trends within the time series itself. Prophet (D) can handle seasonality and holidays effectively, but CNN-QR is generally superior in incorporating additional complex external variables and capturing non-linear dependencies in the time series data.
Therefore, CNN-QR's capacity to handle auxiliary data, capture complex non-linear relationships, and provide probabilistic forecasts makes it the most suitable algorithm for the given energy consumption forecasting task.For more information, you can refer to the Amazon Forecast documentation:https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-algorithms.html and specifically the CNN-QR algorithm section.

Question: 160

Exam Heist

A company wants to use automatic speech recognition (ASR) to transcribe messages that are less than 60 seconds long from a voicemail-style application. The company requires the correct identification of 200 unique product names, some of which have unique spellings or pronunciations. The company has 4,000 words of Amazon SageMaker Ground Truth voicemail transcripts it can use to customize the chosen ASR model. The company needs to ensure that everyone can update their customizations multiple times each hour.
Which approach will maximize transcription accuracy during the development phase?

- Use a voice-driven Amazon Lex bot to perform the ASR customization. Create customer slots within the bot that specifically identify each of the required product names. Use the Amazon Lex synonym mechanism to provide additional variations of each product name as mis-transcriptions are identified in development.
- Use Amazon Transcribe to perform the ASR customization. Analyze the word confidence scores in the transcript, and automatically create or update a custom vocabulary file with any word that has a confidence score below an acceptable threshold value. Use this updated custom vocabulary file in all future transcription tasks.
- Create a custom vocabulary file containing each product name with phonetic pronunciations, and use it with Amazon Transcribe to perform the ASR customization. Analyze the transcripts and manually update the custom vocabulary file to include updated or additional entries for those names that are not being correctly identified. ✓
- Use the audio transcripts to create a training dataset and build an Amazon Transcribe custom language model. Analyze the transcripts and updatethe training dataset with a manually corrected version of transcripts where product names are not being transcribed correctly. Create an updated custom language model.

Explanation:
Here's a detailed justification for why option C is the best approach for maximizing transcription accuracy in this scenario:
The core requirement is accurate transcription of 200 unique product names, some with unusual spellings and pronunciations, using ASR for short voicemail messages. The company has a limited dataset of 4,000 words and needs frequent customization updates.
Option C leverages Amazon Transcribe's custom vocabulary feature, which is specifically designed to improve accuracy for out-of-vocabulary words or words with non-standard pronunciations. By creating a custom vocabulary file containing each product name along with its phonetic pronunciation, the ASR model is guided to correctly identify these specific terms. The iterative approach of analyzing transcripts and manually updating the vocabulary ensures continuous improvement based on real-world transcription performance. This direct control over the vocabulary and pronunciations offers the most targeted and efficient way to address the accuracy needs given the limited dataset. It is feasible to make frequent updates as required.
Option A, using Amazon Lex, is less suitable because Lex is primarily designed for building conversational interfaces, not optimized for general ASR tasks like transcribing voicemail. While it can handle slots and synonyms, its ASR engine might not be as robust as Amazon Transcribe's for this particular scenario. Lex is generally used when you need a bot that can understand and respond to user input.
Option B suggests an automated approach of using word confidence scores to update the custom vocabulary. While appealing, this is risky. Low confidence scores can arise from various factors, not just incorrect pronunciation or vocabulary. Automatically adding words based solely on low confidence might introduce errors and degrade the overall accuracy. There will be manual effort required to review the changes and ensure that there are no mistakes.
Option D, building a custom language model, is the most resource-intensive option. Custom language models require significantly more training data than a custom vocabulary. Given the limited dataset of 4,000 words, building a robust custom language model is unlikely to be effective. The time and effort required to create and maintain a language model would be disproportionate to the likely accuracy gains. Additionally, language model training generally takes more time than updating the custom vocabulary.In summary, option C provides the optimal balance of targeted customization, efficiency with limited data, and feasibility for frequent updates, making it the best choice for maximizing transcription accuracy.
References:

Amazon Transcribe Custom Vocabulary
Amazon Transcribe

Question: 161

Exam Heist

A company is building a demand forecasting model based on machine learning (ML). In the development stage, an ML specialist uses an Amazon SageMaker notebook to perform feature engineering during work hours that consumes low amounts of CPU and memory resources. A data engineer uses the same notebook to perform data preprocessing once a day on average that requires very high memory and completes in only 2 hours. The data preprocessing is not configured to use GPU. All the processes are running well on an ml.m5.4xlarge notebook instance.
The company receives an AWS Budgets alert that the billing for this month exceeds the allocated budget.
Which solution will result in the MOST cost savings?

- Change the notebook instance type to a memory optimized instance with the same vCPU number as the ml.m5.4xlarge instance has. Stop the notebook when it is not in use. Run both data preprocessing and feature engineering development on that instance.
- Keep the notebook instance type and size the same. Stop the notebook when it is not in use. Run data preprocessing on a P3 instance type with the same memory as the ml.m5.4xlarge instance by using Amazon SageMaker Processing.
- Change the notebook instance type to a smaller general purpose instance. Stop the notebook when it is not in use. Run data preprocessing on an ml.r5 instance with the same memory size as the ml.m5.4xlarge instance by using Amazon SageMaker Processing. ✓
- Change the notebook instance type to a smaller general purpose instance. Stop the notebook when it is not in use. Run data preprocessing on an R5 instance with the same memory size as the ml.m5.4xlarge instance by using the Reserved Instance option.

Explanation:
The most cost-effective solution is **C**. Here's why:

Problem: The ml.m5.4xlarge instance is oversized for feature engineering, which has low CPU and memory requirements. The high memory usage is only needed for 2 hours a day for data preprocessing. Running the oversized instance continuously leads to unnecessary costs.
Solution A (Incorrect): Switching to a memory-optimized instance of the same vCPU count doesn't address the core issue of over-provisioning for the majority of the time (feature engineering). The cost is still high as ml.m5.4xlarge and ml.r5.4xlarge will be similarly priced.
Solution B (Incorrect): While stopping the notebook helps, running data preprocessing on a P3 (GPU) instance is wasteful since the preprocessing task is explicitly stated not to use GPU. P3 instances are significantly more expensive than other instance types, as they provide GPU capacity.
Solution C (Correct): This option effectively separates the workloads and optimizes resource utilization. It shrinks the notebook instance to a smaller, general-purpose type suitable for feature engineering (saving costs). It then offloads the memory-intensive data preprocessing to a separate ml.r5 instance using SageMaker Processing. SageMaker Processing allows you to run the data preprocessing script only when needed, and the ml.r5 instance is automatically terminated after the job is complete. This ensures you are only paying for the high-memory instance when it's actively used.
Solution D (Incorrect): Using Reserved Instances (RIs) for an R5 instance might seem cheaper in the long run if the data preprocessing job ran 24/7. However, we know this data preprocessing job only runs for 2 hours/day, which makes reserved instances inefficient and more expensive. Reserved Instances work best for workloads with consistent utilization. Also this option is using the R5 instance directly (implying keeping it up) instead of SageMaker Processing.
Using SageMaker Processing is key because it allows on-demand, temporary resource allocation.
Key Concepts and Justification Points:

Right-Sizing: Matching instance types to actual workload requirements for cost optimization.
https://aws.amazon.com/ec2/instance-types/
SageMaker Processing: A fully managed service for running data processing workloads.
https://aws.amazon.com/sagemaker/processing/
Cost Optimization: Reducing unnecessary spending on cloud resources.
Workload Separation: Isolating different workloads (feature engineering vs. data preprocessing) to optimize resource allocation.
Reserved Instances: Commitment-based pricing model for predictable workloads.
https://aws.amazon.com/ec2/pricing/reserved-instances/
By using a smaller notebook instance for feature engineering and SageMaker Processing with an ml.r5 instance for data preprocessing, the company minimizes resource consumption and only pays for the resources needed when they are actively in use. This results in the most cost savings compared to the other options.

Question: 162

Exam Heist

A machine learning specialist is developing a regression model to predict rental rates from rental listings. A variable named Wall_Color represents the most prominent exterior wall color of the property. The following is the sample data, excluding all other variables:

Property_ID	Wall_Color
1000	Red
1001	White
1002	Green

The specialist chose a model that needs numerical input data.
Which feature engineering approaches should the specialist use to allow the regression model to learn from the Wall_Color data? (Choose two.)

- [ ] Apply integer transformation and set Red = 1, White = 5, and Green = 10.
- [x] Add new columns that store one-hot representation of colors. ✓
- [ ] Replace the color name string by its length.
- [ ] Create three columns to encode the color in RGB format.
- [x] Replace each color name by its training set frequency. ✓

Explanation:
B. Add new columns that store one-hot representation of colors.One-hot encoding is a common approach to represent categorical variables as numerical values. This approach creates new binary variables for each category and assigns a value of 1 to the corresponding category and 0 to the others. In this case, the specialist can create three new binary variables, one for each color (Red, White, and Green) and use them as input to the regression model.E. Replace each color name by its training set frequency.Another approach to convert categorical variables into numerical ones is to replace each category with its frequency of occurrence in the training set. In this case, the specialist can replace the color names with their respective frequencies (1/3 for Red, 1/3 for White, and 1/3 for Green) to represent them numerically.

Question: 163

Exam Heist

A data scientist is working on a public sector project for an urban traffic system. While studying the traffic patterns, it is clear to the data scientist that the traffic behavior at each light is correlated, subject to a small stochastic error term. The data scientist must model the traffic behavior to analyze the traffic patterns and reduce congestion.
How will the data scientist MOST effectively model the problem?

- ○ The data scientist should obtain a correlated equilibrium policy by formulating this problem as a multi-agent reinforcement learning problem. ✓
- ○ The data scientist should obtain the optimal equilibrium policy by formulating this problem as a single-agent reinforcement learning problem.
- ○ Rather than finding an equilibrium policy, the data scientist should obtain accurate predictors of traffic flow by using historical data through a supervised learning approach.
- ○ Rather than finding an equilibrium policy, the data scientist should obtain accurate predictors of traffic flow by using unlabeled simulated data representing the new traffic patterns in the city and applying an unsupervised learning approach.

Explanation:
The correct answer is A: The data scientist should obtain a correlated equilibrium policy by formulating this problem as a multi-agent reinforcement learning problem.
Here's why:

Multi-Agent Setting: Traffic lights inherently operate as individual agents within a larger system. Their actions (e.g., light timings) impact the performance of other traffic lights and overall traffic flow. A single-agent approach (option B) cannot capture these interdependencies effectively.
Correlated Equilibrium: In a system of interdependent agents, a correlated equilibrium is a probability distribution over joint actions that incentivizes each agent to follow its prescribed action, given that the other agents also follow theirs. Finding this equilibrium aims to coordinate the traffic lights to achieve better overall traffic flow and reduce congestion. Each traffic light is correlated with other lights and makes decisions based on the decisions of other lights.
Reinforcement Learning (RL): RL is well-suited for this problem because the system's dynamics are complex and difficult to model directly. RL allows the agents (traffic lights) to learn optimal policies through trial and error, by interacting with the environment (the traffic system) and receiving rewards (e.g., reduced congestion, shorter travel times).
Supervised and Unsupervised Learning Limitations: Supervised learning (option C) could predict traffic flow based on historical data. However, it wouldn't necessarily optimize traffic light control to actively *reduce* congestion. It would simply predict based on existing patterns. Unsupervised learning (option D) is less applicable because it deals with uncovering

patterns in unlabeled data, but here, the goal is to actively control the traffic system, not simply describe the system's properties.

Why Correlated Equilibrium is better than Nash Equilibrium In a Nash equilibrium, each agent chooses its best action independently, assuming the other agents' actions are fixed. In a correlated equilibrium, a central coordinator recommends an action to each agent, and each agent chooses their best action *given* the recommendation. Correlated equilibrium is better here since it can increase the overall traffic flow by ensuring the traffic lights coordinate better with each other.

In essence, a multi-agent reinforcement learning approach focusing on correlated equilibrium allows the traffic lights to learn coordinated policies that optimize the overall traffic system, leading to reduced congestion and improved traffic flow. This is a more proactive and adaptive solution than simply predicting traffic flow based on historical or simulated data.

Authoritative Links:

Multi-Agent Reinforcement Learning (MARL):

A Survey of Multi-Agent Reinforcement Learning: https://arxiv.org/abs/1706.05264

Correlated Equilibrium:

Robert Aumann (1974). "Subjectivity and Correlation in Randomized Strategies". *Journal of Mathematical Economics*. **1** (1): 67–96.

Question: 164

Exam Heist

A data scientist is using the Amazon SageMaker Neural Topic Model (NTM) algorithm to build a model that recommends tags from blog posts. The raw blog post data is stored in an Amazon S3 bucket in JSON format. During model evaluation, the data scientist discovered that the model recommends certain stopwords such as "a," "an," and "the" as tags to certain blog posts, along with a few rare words that are present only in certain blog entries. After a few iterations of tag review with the content team, the data scientist notices that the rare words are unusual but feasible. The data scientist also must ensure that the tag recommendations of the generated model do not include the stopwords.

What should the data scientist do to meet these requirements?

- Use the Amazon Comprehend entity recognition API operations. Remove the detected words from the blog post data. Replace the blog post data source in the S3 bucket.
- Run the SageMaker built-in principal component analysis (PCA) algorithm with the blog post data from the S3 bucket as the data source. Replace the blog post data in the S3 bucket with the results of the training job.
- Use the SageMaker built-in Object Detection algorithm instead of the NTM algorithm for the training job to process the blog post data.
- Remove the stopwords from the blog post data by using the CountVectorizer function in the scikit-learn library. Replace the blog post data in the S3 bucket with the results of the vectorizer. ✓

Explanation:

The correct answer is D. Here's why:

The core problem is that the NTM model is recommending stopwords and rare words as tags, which is undesirable. To address this, we need to preprocess the data to remove these unwanted terms before training the model.

Option D directly addresses this issue by suggesting the use of `CountVectorizer` from the scikit-learn library. `CountVectorizer` allows us to filter out stopwords by specifying a `stop_words` parameter. This allows for customization and integration into a SageMaker pipeline. Removing these common words helps the model focus on more meaningful terms. Moreover, `CountVectorizer` can also filter out terms based on document frequency using `min_df` and `max_df` arguments, which can help reduce the effect of rare words if necessary. By replacing the original data in S3 with the vectorized (and cleaned) data, the NTM model will be trained on preprocessed text, leading to better tag recommendations.

Option A is incorrect. Amazon Comprehend is useful for entity recognition, but it doesn't directly address the problem of stopwords and rare words in the context of topic modeling. Removing entities detected by Comprehend would be too aggressive and could remove important concepts beyond just stopwords.

Option B is incorrect. Principal Component Analysis (PCA) is a dimensionality reduction technique, not a text preprocessing method. It wouldn't help remove stopwords or rare words. It is more relevant for feature engineering in numerical data contexts.

Option C is incorrect. Object Detection is completely unrelated to topic modeling and text analysis. It's designed for identifying objects within images or videos, not extracting relevant tags from text documents. NTM is appropriate for topic modeling.

In summary, option D provides the most direct and relevant approach to resolving the issue of unwanted words in the NTM model's tag recommendations by using established text processing techniques with `CountVectorizer` before training on SageMaker.

For further reading:

scikit-learn CountVectorizer: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Amazon SageMaker NTM Algorithm: https://docs.aws.amazon.com/sagemaker/latest/dg/ntm.html

Question: 165

Exam Heist

A company wants to create a data repository in the AWS Cloud for machine learning (ML) projects. The company wants to use AWS to perform complete ML lifecycles and wants to use Amazon S3 for the data storage. All of the company's data currently resides on premises and is 40 ¡¢¡' in size.
The company wants a solution that can transfer and automatically update data between the on-premises object storage and Amazon S3. The solution must support encryption, scheduling, monitoring, and data integrity validation.
Which solution meets these requirements?

- Use the S3 sync command to compare the source S3 bucket and the destination S3 bucket. Determine which source files do not exist in the destination S3 bucket and which source files were modified.
- Use AWS Transfer for FTPS to transfer the files from the on-premises storage to Amazon S3.
- Use AWS DataSync to make an initial copy of the entire dataset. Schedule subsequent incremental transfers of changing data until the final cutover from on premises to AWS. ✓
- Use S3 Batch Operations to pull data periodically from the on-premises storage. Enable S3 Versioning on the S3 bucket to protect against accidental overwrites.

Explanation:
The correct answer is C, using AWS DataSync. Here's why:

Data Volume and Requirements: The company has a large dataset (40 TB) and requires automated data transfer, updates, encryption, scheduling, monitoring, and data integrity validation.
AWS DataSync: AWS DataSync is a data transfer service designed to move large amounts of data between on-premises storage and AWS storage services like S3. It handles incremental transfers efficiently, only copying changed data after the initial bulk transfer. It inherently provides encryption during transfer and at rest and validates data integrity. DataSync also offers scheduling and monitoring capabilities. https://aws.amazon.com/datasync/
Why other options are less suitable:

A: S3 Sync: While `aws s3 sync` can transfer data, it lacks built-in scheduling, monitoring, encryption, and robust data integrity validation mechanisms suitable for large-scale, automated data migration. It requires custom scripting for many of the required features.
B: AWS Transfer for FTPS: AWS Transfer for FTPS is specifically for FTPS transfers. It does not address large data volumes, scheduled syncing or data integrity validation in a direct way. It is also less suitable for object storage.
D: S3 Batch Operations: S3 Batch Operations operates on objects *already* in S3. It's not a data transfer service for moving data *into* S3 from on-premises storage. It can't pull data from on-premises storage directly. S3 Versioning helps with accidental overwrites but doesn't address the core requirements of automated, secure, and validated data transfer from on-premises. DataSync provides a fully managed service tailored to meet the prompt's specific needs for migrating and synchronizing large datasets to S3. The initial copy followed by scheduled incremental updates is a common best practice for on-premises to cloud migration.

Question: 166

Exam Heist

A company has video feeds and images of a subway train station. The company wants to create a deep learning model that will alert the station manager if any passenger crosses the yellow safety line when there is no train in the station. The alert will be based on the video feeds. The company wants the model to detect the yellow line, the passengers who cross the yellow line, and the trains in the video feeds. This task requires labeling. The video data must remain confidential.
A data scientist creates a bounding box to label the sample data and uses an object detection model. However, the object detection model cannot clearly demarcate the yellow line, the passengers who cross the yellow line, and the trains.
Which labeling approach will help the company improve this model?

- Use Amazon Rekognition Custom Labels to label the dataset and create a custom Amazon Rekognition object detection model. Create a private workforce. Use Amazon Augmented AI (Amazon A2I) to review the low-confidence predictions and retrain the custom Amazon Rekognition model. ✓
- Use an Amazon SageMaker Ground Truth object detection labeling task. Use Amazon Mechanical Turk as the labeling workforce.
- Use Amazon Rekognition Custom Labels to label the dataset and create a custom Amazon Rekognition object detection model. Create a workforce with a third-party AWS Marketplace vendor. Use Amazon Augmented AI (Amazon A2I) to review the low-confidence predictions and retrain the custom Amazon Rekognition model.
- Use an Amazon SageMaker Ground Truth semantic segmentation labeling task. Use a private workforce as the labeling workforce.

Explanation:

Here's a detailed justification for why option A is the most suitable solution, along with explanations of why other options are less optimal.

Justification for Option A: Amazon Rekognition Custom Labels with Private Workforce and A2I

Option A is the best choice because it directly addresses the company's requirements for confidentiality, improved object demarcation, and model accuracy.

Confidentiality: Using Amazon Rekognition Custom Labels allows the company to keep the video data within AWS, avoiding the need to share sensitive information with external services. Creating a private workforce ensures that only trusted individuals have access to the data.

Improved Object Demarcation: While the question mentions initial use of bounding boxes which is characteristic of object detection, the problem states that the objects (yellow line, passengers, trains) are not clearly demarcated, hence the choice for semantic segmentation in option D. However, A custom Amazon Rekognition object detection model still addresses the issue given that it utilizes more refined labeling data.

Model Accuracy: Amazon Augmented AI (A2I) enables human review of low-confidence predictions, which is crucial for improving the accuracy of the object detection model. By retraining the model with the corrected labels, the company can iteratively enhance its performance.

Scalability & Integration: Amazon Rekognition seamlessly integrates with other AWS services, which is advantageous for building a scalable and maintainable solution.

Why Other Options are Less Optimal:

Option B: While Amazon SageMaker Ground Truth is a valid labeling service, using Amazon Mechanical Turk poses a significant risk to the confidentiality of the video data, as the data would be exposed to a public workforce. This directly violates the company's requirements.

Option C: Using a third-party AWS Marketplace vendor for the workforce, while potentially offering specialized skills, also introduces a confidentiality concern. The company would need to carefully vet the vendor and establish strict data security agreements, which can be complex. It does however, provide Augmented AI functionality for the model to improve with data.

Option D: This option suggests using semantic segmentation, which is a good idea, but only using Amazon SageMaker Ground Truth, which could compromise the confidentiality of the video data, as the data would be exposed to a public workforce.

Authoritative Links for Further Research:

Amazon Rekognition Custom Labels: https://aws.amazon.com/rekognition/custom-labels-features/
Amazon SageMaker Ground Truth: https://aws.amazon.com/sagemaker/groundtruth/
Amazon Augmented AI (A2I): https://aws.amazon.com/augmented-ai/

In summary, option A provides the best balance of data confidentiality, object demarcation improvement, and model accuracy enhancement, making it the most suitable solution for the company's requirements.

Question: 167

Exam Heist

A data engineer at a bank is evaluating a new tabular dataset that includes customer data. The data engineer will use the customer data to create a new model to predict customer behavior. After creating a correlation matrix for the variables, the data engineer notices that many of the 100 features are highly correlated with each other.

Which steps should the data engineer take to address this issue? (Choose two.)

- [ ] Use a linear-based algorithm to train the model.
- [x] Apply principal component analysis (PCA). ✓
- [x] Remove a portion of highly correlated features from the dataset. ✓
- [ ] Apply min-max feature scaling to the dataset.
- [ ] Apply one-hot encoding category-based variables.

Explanation:

The data engineer needs to address multicollinearity in the dataset, caused by the high correlation between features. Here's why options B and C are the most appropriate:

B. Apply principal component analysis (PCA). PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated features called principal components. These components capture the most variance in the data, effectively reducing multicollinearity. By using a smaller number of principal components, the data engineer can simplify the model, reduce overfitting, and improve its interpretability. PCA is particularly useful when it's difficult to determine which correlated features to remove. AWS services like SageMaker support PCA through built-in algorithms.

C. Remove a portion of highly correlated features from the dataset. This approach directly tackles multicollinearity by eliminating redundant information. If two features are highly correlated, they likely provide similar information to the model. Removing one of them simplifies the model without significantly impacting its performance. The data engineer should choose the feature to remove based on domain knowledge, missing values, or importance to the target variable. For example, if "annual income" and "credit score" are highly correlated, the engineer might choose to keep "annual income" if it's considered more fundamentally important to predicting customer behavior.
Let's analyze why the other options are less suitable:

A. Use a linear-based algorithm to train the model. While linear models are affected by multicollinearity, simply choosing a linear model doesn't address the underlying issue. Multicollinearity can still lead to unstable coefficient estimates and make it difficult to interpret the feature importance. It's better to deal with the multicollinearity directly through feature engineering (like PCA or feature removal) before training any model.
D. Apply min-max feature scaling to the dataset. Feature scaling (like min-max) is a useful preprocessing step, especially for algorithms sensitive to feature scales (like neural networks or k-nearest neighbors). However, feature scaling does not address multicollinearity. It only rescales the range of feature values.
E. Apply one-hot encoding category-based variables. One-hot encoding is important for handling categorical features by converting them into numerical representations. This is a separate concern from multicollinearity. It's still necessary to address the multicollinearity among numerical features *after* handling categorical features through one-hot encoding.
In summary, PCA and feature removal directly mitigate multicollinearity by reducing the redundancy in the dataset, improving model stability and interpretability within the AWS Machine Learning environment.
Relevant Links:

Principal Component Analysis: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Multicollinearity: https://en.wikipedia.org/wiki/Multicollinearity

Question: 168

Exam Heist

A company is building a new version of a recommendation engine. Machine learning (ML) specialists need to keep adding new data from users to improve personalized recommendations. The ML specialists gather data from the users' interactions on the platform and from sources such as external websites and social media.
The pipeline cleans, transforms, enriches, and compresses terabytes of data daily, and this data is stored in Amazon S3. A set of Python scripts was coded to do the job and is stored in a large Amazon EC2 instance. The whole process takes more than 20 hours to finish, with each script taking at least an hour. The company wants to move the scripts out of Amazon EC2 into a more managed solution that will eliminate the need to maintain servers.
Which approach will address all of these requirements with the LEAST development effort?

- Load the data into an Amazon Redshift cluster. Execute the pipeline by using SQL. Store the results in Amazon S3.
- Load the data into Amazon DynamoDB. Convert the scripts to an AWS Lambda function. Execute the pipeline by triggering Lambda executions. Store the results in Amazon S3.
- Create an AWS Glue job. Convert the scripts to PySpark. Execute the pipeline. Store the results in Amazon S3. ✓
- Create a set of individual AWS Lambda functions to execute each of the scripts. Build a step function by using the AWS Step Functions Data Science SDK. Store the results in Amazon S3.

Explanation:
The best approach is **C. Create an AWS Glue job. Convert the scripts to PySpark. Execute the pipeline. Store the results in Amazon S3.**
Here's why:

Managed Solution: AWS Glue is a fully managed extract, transform, and load (ETL) service, eliminating the need to manage servers, fulfilling the core requirement.
Scalability and Performance: Glue with Spark offers inherent scalability and distributed processing, crucial for handling terabytes of data efficiently. PySpark optimizes for large datasets.
Data Handling: Glue is designed for data transformation in S3, fitting the scenario perfectly. It can read, transform, and write data directly to S3.
Reduced Development Effort: Refactoring the existing Python scripts to PySpark is less effort than rewriting them as individual Lambda functions or SQL queries for Redshift. Glue's visual interface also simplifies job definition.
Let's analyze why the other options are less suitable:

A (Redshift): While Redshift handles large datasets, it's primarily a data warehouse for analytical queries, not an ETL engine for complex transformations from multiple sources. Converting complex Python scripts to SQL would be significantly more development effort.

B (DynamoDB and Lambda): DynamoDB is a NoSQL database, not suitable for directly ingesting and transforming terabytes of data from diverse sources. Converting complex scripts to Lambda functions and orchestrating them would be highly complex and less efficient for data transformation.
D (Lambda and Step Functions): Using Lambda for individual scripts and Step Functions for orchestration is a valid approach, but less efficient and more complex than Glue. Lambda has limitations in runtime and memory, especially with the size of data needed to be processed. This would require splitting scripts and handling state management, increasing development time.
In summary: AWS Glue offers a fully managed, scalable, and efficient solution for transforming large datasets stored in S3 with minimal development effort by leveraging the familiarity of python and the ease of Spark on Glue.
Relevant Links:

AWS Glue: https://aws.amazon.com/glue/
PySpark: https://spark.apache.org/docs/latest/api/python/

Question: 169

Exam Heist

A retail company is selling products through a global online marketplace. The company wants to use machine learning (ML) to analyze customer feedback and identify specific areas for improvement. A developer has built a tool that collects customer reviews from the online marketplace and stores them in an Amazon S3 bucket. This process yields a dataset of 40 reviews. A data scientist building the ML models must identify additional sources of data to increase the size of the dataset.
Which data sources should the data scientist use to augment the dataset of reviews? (Choose three.)

- [x] Emails exchanged by customers and the company's customer service agents ✓
- [x] Social media posts containing the name of the company or its products ✓
- [ ] A publicly available collection of news articles
- [x] A publicly available collection of customer reviews ✓
- [ ] Product sales revenue figures for the company
- [ ] Instruction manuals for the company's products

Explanation:
The correct answer is ABD because these options directly provide textual data that can be used to understand customer sentiment and areas for improvement. Let's analyze each choice:

A. Emails exchanged by customers and the company's customer service agents: These emails represent direct customer communication. They contain valuable insights into customer issues, complaints, and satisfaction levels, making them a rich source of data for sentiment analysis and identifying areas for improvement. This data reflects real-world customer experiences and opinions related to the company's products or services.
B. Social media posts containing the name of the company or its products: Social media platforms are breeding grounds for public opinion. Analyzing social media posts can reveal what customers are saying about the company, its products, and its services. This helps gauge overall sentiment, identify trending topics, and uncover potential issues that the company might not be aware of. This is a readily available, public source of customer feedback.
D. A publicly available collection of customer reviews: Augmenting the initial dataset of 40 reviews with a larger, publicly available collection of customer reviews significantly increases the size and diversity of the data. More data typically leads to more robust and accurate machine learning models. Public reviews are often available on websites like Amazon, Yelp, and Google Reviews.
Why C, E, and F are incorrect:

C. A publicly available collection of news articles: News articles may mention the company, but they are unlikely to reflect direct customer feedback or opinions on specific products. This data would be less relevant to the task of analyzing customer sentiment and identifying areas for improvement.
E. Product sales revenue figures for the company: Sales figures provide information about product performance but offer no insight into customer sentiment, satisfaction, or specific areas for improvement based on customer experiences.
F. Instruction manuals for the company's products: Instruction manuals are informative documents but do not contain customer feedback or sentiment. They are irrelevant to the task of analyzing customer opinions.
Therefore, the email correspondence, social media posts, and external customer reviews are the most effective data sources for augmenting the dataset and improving the performance of sentiment analysis models focused on customer experience. The additional textual data will enrich the model's ability to understand customer feedback and identify areas for improvement.
Further Research:

Sentiment Analysis: https://monkeylearn.com/sentiment-analysis/

Text Data Augmentation: https://neptune.ai/blog/data-augmentation-nlp

Question: 170

Exam Heist

A machine learning (ML) specialist wants to create a data preparation job that uses a PySpark script with complex window aggregation operations to create data for training and testing. The ML specialist needs to evaluate the impact of the number of features and the sample count on model performance.
Which approach should the ML specialist use to determine the ideal data transformations for the model?

- Add an Amazon SageMaker Debugger hook to the script to capture key metrics. Run the script as an AWS Glue job.
- Add an Amazon SageMaker Experiments tracker to the script to capture key metrics. Run the script as an AWS Glue job.
- Add an Amazon SageMaker Debugger hook to the script to capture key parameters. Run the script as a SageMaker processing job.
- Add an Amazon SageMaker Experiments tracker to the script to capture key parameters. Run the script as a SageMaker processing job. ✓

Explanation:
The correct approach is to use Amazon SageMaker Experiments to track key parameters and run the script as a SageMaker processing job. SageMaker Experiments is designed for managing, tracking, and comparing machine learning experiments. It allows you to record input parameters (number of features, sample count), metrics (model performance), and artifacts produced during each run. This capability directly addresses the requirement to evaluate the impact of different features and sample counts on model performance. A SageMaker processing job is designed to execute data preprocessing, feature engineering, and model evaluation tasks. It provides a managed environment for running your PySpark script and integrates seamlessly with SageMaker Experiments.
Options A and C are incorrect because SageMaker Debugger is primarily used for debugging training jobs, specifically to identify and diagnose training issues like vanishing gradients or exploding weights. Debugger is not ideal for tracking and comparing the impact of different data transformations on model performance. Option B is incorrect because while it uses SageMaker Experiments, it suggests using AWS Glue. Although Glue can run PySpark scripts, it's primarily designed for ETL (extract, transform, load) tasks. SageMaker Processing jobs offer better integration with SageMaker services like Experiments, and are intended for ML-specific workloads. Using SageMaker processing job directly integrates and stores relevant metadata within the SageMaker environment.
In essence, SageMaker Experiments + Processing job is the ideal combination for managing and tracking your data preparation experiments, while Debugger focuses on model training debugging. Glue is suitable for ETL, but not as tightly coupled with ML experimentation compared to Processing Jobs.
References:

Amazon SageMaker Experiments: https://aws.amazon.com/sagemaker/experiments/
Amazon SageMaker Processing Jobs: https://aws.amazon.com/sagemaker/processing-jobs/
Amazon SageMaker Debugger: https://aws.amazon.com/sagemaker/debugger/
AWS Glue: https://aws.amazon.com/glue/

Question: 171

Exam Heist

A data scientist has a dataset of machine part images stored in Amazon Elastic File System (Amazon EFS). The data scientist needs to use Amazon SageMaker to create and train an image classification machine learning model based on this dataset. Because of budget and time constraints, management wants the data scientist to create and train a model with the least number of steps and integration work required.
How should the data scientist meet these requirements?

- Mount the EFS file system to a SageMaker notebook and run a script that copies the data to an Amazon FSx for Lustre file system. Run the SageMaker training job with the FSx for Lustre file system as the data source.
- Launch a transient Amazon EMR cluster. Configure steps to mount the EFS file system and copy the data to an Amazon S3 bucket by using S3DistCp. Run the SageMaker training job with Amazon S3 as the data source.
- Mount the EFS file system to an Amazon EC2 instance and use the AWS CLI to copy the data to an Amazon S3 bucket. Run the SageMaker training job with Amazon S3 as the data source.
- Run a SageMaker training job with an EFS file system as the data source. ✓

Explanation:
The best way to address the data scientist's requirements of minimal steps, integration effort, budget, and time constraints while using data stored in EFS for a SageMaker training job is option D: Run a SageMaker training job with an EFS file system as the data source.

Here's why:

Direct Integration: SageMaker directly supports EFS as a data source for training jobs. This eliminates the need for intermediate data copying or transformations.
Reduced Complexity: Options A, B, and C all involve extra steps of copying data to FSx for Lustre or S3. These steps add complexity, increase the time required for training setup, and potentially incur additional costs (FSx for Lustre is more expensive than S3 for storage, and EMR cluster brings its own costs).
Minimized Costs: Copying data requires resources (compute, storage, network) and time, which translates to increased costs. Using EFS directly minimizes these costs.
Faster Time to Model: By avoiding data copying, the data scientist can quickly initiate the training job and reduce the overall time required to build and train the model.
Options A, B, and C are valid solutions in some contexts, but they do not satisfy the constraint of minimizing steps and integration work. For instance, Amazon FSx for Lustre (Option A) can speed up training in some cases by providing a high performance file system, but is not necessary here given the specific requirements. Likewise using EMR or EC2 Instances just to move the data to S3 or another source adds extra overhead and complexity.
Therefore, using SageMaker's direct integration with EFS for training jobs is the most efficient and cost-effective approach.
Relevant Documentation:

Using Amazon EFS file systems with Amazon SageMaker

Question: 172

Exam Heist

A retail company uses a machine learning (ML) model for daily sales forecasting. The company's brand manager reports that the model has provided inaccurate results for the past 3 weeks.
At the end of each day, an AWS Glue job consolidates the input data that is used for the forecasting with the actual daily sales data and the predictions of the model. The AWS Glue job stores the data in Amazon S3. The company's ML team is using an Amazon SageMaker Studio notebook to gain an understanding about the source of the model's inaccuracies.
What should the ML team do on the SageMaker Studio notebook to visualize the model's degradation MOST accurately?

- Create a histogram of the daily sales over the last 3 weeks. In addition, create a histogram of the daily sales from before that period.
- Create a histogram of the model errors over the last 3 weeks. In addition, create a histogram of the model errors from before that period.
- Create a line chart with the weekly mean absolute error (MAE) of the model. ✓
- Create a scatter plot of daily sales versus model error for the last 3 weeks. In addition, create a scatter plot of daily sales versus model error from before that period.

Explanation:
The most accurate way to visualize model degradation in this scenario is to track the trend of the model's error over time. Option C suggests creating a line chart of the weekly mean absolute error (MAE). This visualization provides a clear and concise representation of how the model's performance has changed over the past weeks.
MAE is a common metric for evaluating the accuracy of regression models, like sales forecasting models. It represents the average magnitude of errors in a set of predictions, without considering their direction. Tracking the weekly MAE allows the ML team to see if the error is consistently increasing, indicating degradation.
Histograms (options A and B) show the distribution of values but don't directly illustrate the trend of error over time. While histograms of sales or errors might provide some insights, they don't clearly demonstrate how the model's performance has degraded *specifically over the last 3 weeks* relative to its prior performance. Comparing histograms requires visual estimation of the shift, which is less precise than a line chart showing the MAE trend.
Scatter plots (option D) can reveal relationships between sales and errors, but are less effective for visualizing performance trends. They would show if there is a change in relationship between the sales and error but not provide a direct measure of error trends. The interpretation would be more involved.
By visualizing the trend of MAE over time, the team can easily identify when the model started to degrade and how severe the degradation is. This information is crucial for troubleshooting the issue and taking corrective action. For example, if the MAE has been steadily increasing, it suggests a gradual drift in the data or model.
Here are some authoritative links for more information:

Mean Absolute Error (MAE): https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html
Model Performance Evaluation: https://developers.google.com/machine-learning/crash-course/descending-into-ml/evaluating-regression

Question: 173

Exam Heist

An ecommerce company sends a weekly email newsletter to all of its customers. Management has hired a team of writers to create additional targeted content. A data scientist needs to identify five customer segments based on age, income, and location. The customers' current segmentation is unknown. The data scientist previously built an XGBoost model to predict the likelihood of a customer responding to an email based on age, income, and location.
Why does the XGBoost model NOT meet the current requirements, and how can this be fixed?

- The XGBoost model provides a true/false binary output. Apply principal component analysis (PCA) with five feature dimensions to predict a segment.
- The XGBoost model provides a true/false binary output. Increase the number of classes the XGBoost model predicts to five classes to predict a segment.
- The XGBoost model is a supervised machine learning algorithm. Train a k-Nearest-Neighbors (kNN) model with K = 5 on the same dataset to predict a segment.
- The XGBoost model is a supervised machine learning algorithm. Train a k-means model with K = 5 on the same dataset to predict a segment. ✓

Explanation:
The original XGBoost model is designed for a supervised learning task: predicting whether a customer will respond to an email (binary classification). It requires labeled data where we already know the outcome (responded or not). The current requirement is to *discover* customer segments based on age, income, and location, without pre-existing labels for these segments. This is an unsupervised learning problem.
Supervised models like XGBoost cannot inherently identify previously unknown clusters within data. They need examples of what each class (segment) looks like, which are unavailable in this scenario. A supervised model would require pre-defined customer segments to learn from. Applying Principal Component Analysis (PCA), as suggested in option A, while helpful for dimensionality reduction, doesn't directly address the need for segmentation; it would simply reduce the number of features used by XGBoost. Option B suggests modifying the XGBoost model to predict five classes, but this still requires predefined and labeled segments, defeating the purpose of discovering them. Option C suggests k-Nearest Neighbors (kNN), which *can* be used for classification if labelled data is available. However, the need for a *discovery* method in the absence of labeled data remains.
K-means clustering, on the other hand, is an unsupervised learning algorithm specifically designed to partition data into K clusters based on feature similarity. By setting K=5, we instruct k-means to find five distinct customer segments based on the age, income, and location data provided. K-means doesn't require pre-labeled data; it identifies clusters solely based on the inherent structure of the data, making it appropriate for discovering previously unknown customer segments. Therefore, replacing the XGBoost model with a k-means model with K=5 directly addresses the business problem of identifying five customer segments based on the provided customer data.
Relevant links:

K-Means Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
Supervised vs. Unsupervised Learning: https://www.ibm.com/cloud/learn/supervised-learning

Question: 174

Exam Heist

A global financial company is using machine learning to automate its loan approval process. The company has a dataset of customer information. The dataset contains some categorical fields, such as customer location by city and housing status. The dataset also includes financial fields in different units, such as account balances in US dollars and monthly interest in US cents.
The company's data scientists are using a gradient boosting regression model to infer the credit score for each customer. The model has a training accuracy of
99% and a testing accuracy of 75%. The data scientists want to improve the model's testing accuracy.
Which process will improve the testing accuracy the MOST?

- Use a one-hot encoder for the categorical fields in the dataset. Perform standardization on the financial fields in the dataset. Apply L1 regularization to the data. ✓
- Use tokenization of the categorical fields in the dataset. Perform binning on the financial fields in the dataset. Remove the outliers in the data by using the z-score.
- Use a label encoder for the categorical fields in the dataset. Perform L1 regularization on the financial fields in the dataset. Apply L2 regularization to the data.
- Use a logarithm transformation on the categorical fields in the dataset. Perform binning on the financial fields in the dataset. Use imputation to populate missing values in the dataset.

Explanation:

The provided scenario describes a model suffering from significant overfitting, evidenced by the large gap between training and testing accuracy (99% vs. 75%). The goal is to improve the model's generalization ability, leading to higher testing accuracy.
Option A addresses overfitting most effectively. Using a one-hot encoder for categorical features avoids introducing arbitrary ordinal relationships that a label encoder (in option C) would create. This is especially crucial for features like city, where no inherent order exists. Standardization of financial fields ensures all features contribute equally to the model's learning process, preventing features with larger values from dominating. L1 regularization encourages sparsity in the model by driving some feature weights to zero, simplifying the model and reducing overfitting.
Option B's tokenization is more suited to natural language processing, not general categorical features. Binning can lose information and might not always improve accuracy. Outlier removal, while sometimes beneficial, can also remove genuine data points.
Option C's label encoding introduces artificial order in categorical variables and L2 regularization, while helpful, is generally less effective at feature selection than L1, making it less optimal for this specific overfitting case.
Option D's logarithmic transformation is unusual for categorical data. Binning has drawbacks as mentioned, and while imputation is a good practice for missing data, it doesn't directly address the core overfitting issue.
Therefore, option A is the most likely to significantly improve testing accuracy by appropriately handling categorical features, scaling numerical features, and employing a powerful regularization technique specifically targeted at reducing model complexity and overfitting.
Authoritative Links:

One-Hot Encoding vs. Label Encoding: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
StandardScaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
L1 Regularization (Lasso): https://scikit-learn.org/stable/modules/linear_model.html#lasso
Overfitting: https://developers.google.com/machine-learning/crash-course/generalization/peril-of-overfitting

Question: 175

Exam Heist

A machine learning (ML) specialist needs to extract embedding vectors from a text series. The goal is to provide a ready-to-ingest feature space for a data scientist to develop downstream ML predictive models. The text consists of curated sentences in English. Many sentences use similar words but in different contexts. There are questions and answers among the sentences, and the embedding space must differentiate between them. Which options can produce the required embedding vectors that capture word context and sequential QA information? (Choose two.)

- [x] Amazon SageMaker seq2seq algorithm ✓
- [ ] Amazon SageMaker BlazingText algorithm in Skip-gram mode
- [x] Amazon SageMaker Object2Vec algorithm ✓
- [ ] Amazon SageMaker BlazingText algorithm in continuous bag-of-words (CBOW) mode
- [ ] Combination of the Amazon SageMaker BlazingText algorithm in Batch Skip-gram mode with a custom recurrent neural network (RNN)

Explanation:
Here's a breakdown of why options A and C are the most suitable, along with explanations of why the other options are less ideal:
A. Amazon SageMaker seq2seq algorithm

Justification: Seq2Seq models are specifically designed for sequence-to-sequence tasks, making them well-suited for understanding context and relationships within sequential data like sentences. This is crucial for differentiating the meaning of similar words used in varying contexts. Seq2Seq models excel at encoding the input sequence (the sentence) into a fixed-length vector (the embedding) and then decoding it into a different sequence (which, in this case, isn't strictly necessary but helps to capture the sequence's meaning). Importantly, Seq2Seq models can be trained to distinguish between questions and answers by treating them as different categories of sequences during training. The model learns to encode questions and answers into distinct embedding spaces.
Cloud Concept: Leverages managed machine learning services to perform advanced sequence modeling.
Authoritative Link: https://docs.aws.amazon.com/sagemaker/latest/dg/seq2seq.html
C. Amazon SageMaker Object2Vec algorithm

Justification: Object2Vec is designed to learn vector representations (embeddings) for a wide variety of objects, including sentences and even question-answer pairs. A critical advantage is its ability to learn relationships between different types of objects. In this scenario, you can feed the algorithm paired questions and answers. Object2Vec can then be trained to create

embeddings that capture the semantic relationship between a question and its corresponding answer. This is especially useful given that the sentences include questions and answers that must be differentiated. Object2Vec can learn the contextual information necessary to do so.
Cloud Concept: Employs a managed machine learning service offering flexible embedding generation for diverse data types.
Authoritative Link: https://docs.aws.amazon.com/sagemaker/latest/dg/object2vec.html
Why other options are less suitable:

B. Amazon SageMaker BlazingText algorithm in Skip-gram mode: Skip-gram (and CBOW) are word embedding techniques. While useful for generating word embeddings, they typically don't inherently capture the entire sentence context or distinguish between question and answer sentence types effectively without further processing. They primarily focus on the local context of individual words.
D. Amazon SageMaker BlazingText algorithm in continuous bag-of-words (CBOW) mode: CBOW suffers from the same limitations as Skip-gram; it is word-centric and lacks the ability to understand entire sentence meaning and the Q&A relationship needed.
E. Combination of the Amazon SageMaker BlazingText algorithm in Batch Skip-gram mode with a custom recurrent neural network (RNN): This option introduces unnecessary complexity. While an RNN *could* be used after BlazingText to try and capture sentence context, using Seq2Seq or Object2Vec directly is a more straightforward and efficient approach to address the prompt's requirements. It also increases operational overhead for model training and deployment.

Question: 176

Exam Heist

A retail company wants to update its customer support system. The company wants to implement automatic routing of customer claims to different queues to prioritize the claims by category.
Currently, an operator manually performs the category assignment and routing. After the operator classifies and routes the claim, the company stores the claim's record in a central database. The claim's record includes the claim's category.
The company has no data science team or experience in the field of machine learning (ML). The company's small development team needs a solution that requires no ML expertise.
Which solution meets these requirements?

- Export the database to a .csv file with two columns: claim_label and claim_text. Use the Amazon SageMaker Object2Vec algorithm and the .csv file to train a model. Use SageMaker to deploy the model to an inference endpoint. Develop a service in the application to use the inference endpoint to process incoming claims, predict the labels, and route the claims to the appropriate queue.
- Export the database to a .csv file with one column: claim_text. Use the Amazon SageMaker Latent Dirichlet Allocation (LDA) algorithm and the .csv file to train a model. Use the LDA algorithm to detect labels automatically. Use SageMaker to deploy the model to an inference endpoint. Develop a service in the application to use the inference endpoint to process incoming claims, predict the labels, and route the claims to the appropriate queue.
- Use Amazon Textract to process the database and automatically detect two columns: claim_label and claim_text. Use Amazon Comprehend custom classification and the extracted information to train the custom classifier. Develop a service in the application to use the Amazon Comprehend API to process incoming claims, predict the labels, and route the claims to the appropriate queue.
- Export the database to a .csv file with two columns: claim_label and claim_text. Use Amazon Comprehend custom classification and the .csv file to train the custom classifier. Develop a service in the application to use the Amazon Comprehend API to process incoming claims, predict the labels, and route the claims to the appropriate queue. ✓

Explanation:
The correct answer is D. Here's why:
The problem emphasizes the need for a solution requiring no ML expertise. Amazon Comprehend custom classification is designed for this purpose. It allows users to train a classifier with their own data without deep ML knowledge.
Option A is incorrect because SageMaker Object2Vec requires ML expertise for feature engineering, hyperparameter tuning, and model deployment management, contradicting the "no ML expertise" requirement.
Option B is incorrect because SageMaker LDA is an unsupervised learning technique for topic modeling, not classification. It won't predict specific claim categories from pre-existing labels as the requirement outlines. Further, it still involves ML model management.
Option C uses Amazon Textract, which is useful for extracting text from scanned documents, but unnecessary in this case as the claim data is already in a database. While Comprehend Custom Classification is present, the initial Textract step adds unnecessary complexity and cost.
Option D provides the most suitable solution because it directly leverages the existing labeled data (claim_label and claim_text) to train a custom classifier in Amazon Comprehend. Comprehend handles the underlying ML complexities. The development team can easily integrate the Comprehend API into their application to predict labels for new claims and route them accordingly, without extensive ML knowledge. The solution is serverless in nature.
Therefore, option D aligns perfectly with the need for a solution requiring minimal ML expertise while achieving the objective of automated claim routing.

Amazon Comprehend Custom Classification

Question: 177

Exam Heist

A machine learning (ML) specialist is using Amazon SageMaker hyperparameter optimization (HPO) to improve a model's accuracy. The learning rate parameter is specified in the following HPO configuration:

```
{
    "Name": "learning_rate",
    "MaxValue" : "0.0001",
    "MinValue": "0.1"
}
```

During the results analysis, the ML specialist determines that most of the training jobs had a learning rate between 0.01 and 0.1. The best result had a learning rate of less than 0.01. Training jobs need to run regularly over a changing dataset. The ML specialist needs to find a tuning mechanism that uses different learning rates more evenly from the provided range between MinValue and MaxValue.
Which solution provides the MOST accurate result?

- Modify the HPO configuration as follows: Select the most accurate hyperparameter configuration form this HPO job.
- Run three different HPO jobs that use different learning rates form the following intervals for MinValue and MaxValue while using the same number of training jobs for each HPO job: ⇝ [0.01, 0.1] ⇝ [0.001, 0.01] ⇝ [0.0001, 0.001] Select the most accurate hyperparameter configuration form these three HPO jobs.
- Modify the HPO configuration as follows: Select the most accurate hyperparameter configuration form this training job. ✓
- Run three different HPO jobs that use different learning rates form the following intervals for MinValue and MaxValue. Divide the number of training jobs for each HPO job by three: ⇝ [0.01, 0.1] ⇝ [0.001, 0.01] [0.0001, 0.001] Select the most accurate hyperparameter configuration form these three HPO jobs.

Explanation:
"Choose logarithmic scaling when you are searching a range that spans several orders of magnitude. For example, if you are tuning a Tune a linear learner model model, and you specify a range of values between .0001 and 1.0 for the learning_rate hyperparameter, searching uniformly on a logarithmic scale gives you a better sample of the entire range than searching on a linear scale would, because searching on a linear scale would, on average, devote 90 percent of your training budget to only the values between .1 and 1.0, leaving only 10 percent of your training budget for the values between .0001 and .1."
https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-define-ranges.html

Question: 178

Exam Heist

A manufacturing company wants to use machine learning (ML) to automate quality control in its facilities. The facilities are in remote locations and have limited internet connectivity. The company has 20 ¡¢¡' of training data that consists of labeled images of defective product parts. The training data is in the corporate on-premises data center.
The company will use this data to train a model for real-time defect detection in new parts as the parts move on a conveyor belt in the facilities. The company needs a solution that minimizes costs for compute infrastructure and that maximizes the scalability of resources for training. The solution also must facilitate the company's use of an ML model in the low-connectivity environments.
Which solution will meet these requirements?

- Move the training data to an Amazon S3 bucket. Train and evaluate the model by using Amazon SageMaker. Optimize the model by using SageMaker Neo. Deploy the model on a SageMaker hosting services endpoint.
- Train and evaluate the model on premises. Upload the model to an Amazon S3 bucket. Deploy the model on an Amazon SageMaker hosting services endpoint.
- Move the training data to an Amazon S3 bucket. Train and evaluate the model by using Amazon SageMaker. Optimize the model by using SageMaker Neo. Set up an edge device in the manufacturing facilities with AWS IoT Greengrass. Deploy the model on the edge device. ✓
- Train the model on premises. Upload the model to an Amazon S3 bucket. Set up an edge device in the manufacturing facilities with AWS IoT Greengrass. Deploy the model on the edge device.

Explanation:
The optimal solution is C because it addresses all the requirements: minimizing cost, maximizing scalability for training, and enabling model usage in low-connectivity environments.
Here's why:

Training in the Cloud (SageMaker): Moving the data to Amazon S3 and using SageMaker for training provides scalable and cost-effective compute resources. SageMaker handles the infrastructure scaling, so the company doesn't have to manage on-

premises hardware, reducing costs and improving resource utilization. https://aws.amazon.com/sagemaker/
Model Optimization (SageMaker Neo): Optimizing the model with SageMaker Neo makes it efficient for deployment on edge devices with limited resources. Neo compiles the model to run faster and with a smaller footprint. https://aws.amazon.com/sagemaker/neo/
Edge Deployment (AWS IoT Greengrass): AWS IoT Greengrass allows the model to be deployed and run locally on edge devices in the manufacturing facilities. This addresses the low-connectivity requirement by enabling real-time defect detection without constant internet access. https://aws.amazon.com/greengrass/
Why other options are suboptimal:

Option A: Training on SageMaker is good, but deploying the model on a SageMaker hosting service endpoint requires constant internet connectivity, which violates the low-connectivity requirement.
Option B: Training on-premises misses the opportunity to leverage scalable and cost-effective cloud resources. Deploying on a SageMaker endpoint has the same connectivity issue as option A.
Option D: Training on-premises is again a missed opportunity for cloud scalability. Uploading to S3 and deploying on Greengrass is partially correct, but the lack of cloud training and model optimization reduces overall efficiency and scalability.
In conclusion, option C leverages the best of cloud training for scalability and cost, coupled with edge deployment for low-connectivity environments, aligning perfectly with the company's needs.

Question: 179

Exam Heist

A company has an ecommerce website with a product recommendation engine built in TensorFlow. The recommendation engine endpoint is hosted by Amazon
SageMaker. Three compute-optimized instances support the expected peak load of the website.
Response times on the product recommendation page are increasing at the beginning of each month. Some users are encountering errors. The website receives the majority of its traffic between 8 AM and 6 PM on weekdays in a single time zone.
Which of the following options are the MOST effective in solving the issue while keeping costs to a minimum? (Choose two.)

- [x] Configure the endpoint to use Amazon Elastic Inference (EI) accelerators. ✓
- [ ] Create a new endpoint configuration with two production variants.
- [x] Configure the endpoint to automatically scale with the InvocationsPerInstance metric. ✓
- [ ] Deploy a second instance pool to support a blue/green deployment of models.
- [ ] Reconfigure the endpoint to use burstable instances.

Explanation:
Here's a detailed justification for why options A and C are the most effective solutions, along with supporting information:
Option A: Configure the endpoint to use Amazon Elastic Inference (EI) accelerators.

Justification: The increasing response times and errors suggest that the existing instances are struggling to keep up with the computational demands of the TensorFlow model, especially at peak times. Elastic Inference (EI) allows you to attach low-cost GPU-powered acceleration to existing Amazon SageMaker endpoints. Instead of scaling up to larger, more expensive instances, you can augment your existing compute-optimized instances with the precise amount of GPU acceleration they need for inference. This avoids over-provisioning and reduces costs. The model's core logic remains on the existing instance type, while computationally intensive parts of the model are offloaded to the EI accelerator.
Relevant Concepts: Inference acceleration, cost optimization, GPU utilization, and model optimization.
Authoritative Links:

Amazon Elastic Inference
Using Elastic Inference with SageMaker
Option C: Configure the endpoint to automatically scale with the InvocationsPerInstance metric.

Justification: The problem description clearly states the website experiences predictable peak traffic during specific hours and escalating issues at the beginning of each month. Auto Scaling based on the `InvocationsPerInstance` metric allows the SageMaker endpoint to dynamically adjust the number of instances based on the actual load. As the number of requests per instance increases (indicating higher load), Auto Scaling will automatically launch additional instances to handle the traffic. This helps maintain consistent response times and prevent errors during peak periods. Because the load is temporal and known, scheduling autoscaling might make the endpoint even more efficient.
Relevant Concepts: Auto Scaling, load balancing, horizontal scaling, performance monitoring, cloud elasticity.
Authoritative Links:

Automatically Scale Amazon SageMaker Models
Scale Your SageMaker Model Serving with Auto Scaling

SageMaker Endpoint Autoscaling Metrics

Why other options are less effective:

Option B: Create a new endpoint configuration with two production variants. While this could facilitate A/B testing of different model versions, it doesn't directly address the issue of increasing response times and errors due to load.
Option D: Deploy a second instance pool to support a blue/green deployment of models. Blue/green deployment is primarily for updating models with minimal downtime. While beneficial for model deployment strategies, it doesn't solve the immediate problem of current performance issues.
Option E: Reconfigure the endpoint to use burstable instances. Burstable instances (like t3.medium) are designed for workloads with infrequent bursts of activity, not sustained high loads. Since the website has predictable peak traffic, burstable instances are not an ideal solution and will likely lead to performance degradation and throttling once the burst credits are exhausted. Compute optimized instances, or augmented standard ones, are generally recommended.
In summary, combining EI for GPU acceleration and Auto Scaling based on `InvocationsPerInstance` provides the most cost-effective and efficient solution to address the performance issues of the recommendation engine.

Question: 180

Exam Heist

A real-estate company is launching a new product that predicts the prices of new houses. The historical data for the properties and prices is stored in .csv format in an Amazon S3 bucket. The data has a header, some categorical fields, and some missing values. The company's data scientists have used Python with a common open-source library to fill the missing values with zeros. The data scientists have dropped all of the categorical fields and have trained a model by using the open-source linear regression algorithm with the default parameters.
The accuracy of the predictions with the current model is below 50%. The company wants to improve the model performance and launch the new product as soon as possible.
Which solution will meet these requirements with the LEAST operational overhead?

- Create a service-linked role for Amazon Elastic Container Service (Amazon ECS) with access to the S3 bucket. Create an ECS cluster that is based on an AWS Deep Learning Containers image. Write the code to perform the feature engineering. Train a logistic regression model for predicting the price, pointing to the bucket with the dataset. Wait for the training job to complete. Perform the inferences.
- Create an Amazon SageMaker notebook with a new IAM role that is associated with the notebook. Pull the dataset from the S3 bucket. Explore different combinations of feature engineering transformations, regression algorithms, and hyperparameters. Compare all the results in the notebook, and deploy the most accurate configuration in an endpoint for predictions.
- Create an IAM role with access to Amazon S3, Amazon SageMaker, and AWS Lambda. Create a training job with the SageMaker built-in XGBoost model pointing to the bucket with the dataset. Specify the price as the target feature. Wait for the job to complete. Load the model artifact to a Lambda function for inference on prices of new houses.
- Create an IAM role for Amazon SageMaker with access to the S3 bucket. Create a SageMaker AutoML job with SageMaker Autopilot pointing to the bucket with the dataset. Specify the price as the target attribute. Wait for the job to complete. Deploy the best model for predictions. ✓

Explanation:
The best solution to improve the real-estate prediction model with the least operational overhead is option D, leveraging SageMaker Autopilot. Here's why:

SageMaker Autopilot: Autopilot automates the entire machine learning pipeline, from data exploration and feature engineering to model selection and hyperparameter tuning. It drastically reduces the manual effort required for model development and deployment. https://aws.amazon.com/sagemaker/autopilot/
Reduced Manual Effort: Given the company's need for speed, Autopilot removes the need for data scientists to manually explore features, experiment with different algorithms, and tune hyperparameters. This aligns with the "least operational overhead" requirement.
Automatic Feature Engineering: Autopilot automatically handles feature engineering, including categorical features and missing values, unlike the current approach of dropping categorical features and filling missing values with zeros. This can significantly improve model accuracy.
Automated Model Selection: Autopilot automatically explores a range of algorithms (including XGBoost) and hyperparameters, and selects the best performing model based on the specified target metric. This ensures better performance than the current linear regression model with default parameters.
Simplified Deployment: Once Autopilot completes, the best model can be easily deployed as a SageMaker endpoint for real-time predictions.
Why other options are less suitable:

Option A (ECS with Deep Learning Containers): Requires significant manual coding for feature engineering and model training, increasing operational overhead. Also, it doesn't automate the model selection and hyperparameter tuning.
Option B (SageMaker Notebook): Requires extensive manual exploration and experimentation within the notebook, making it time-consuming and increasing operational overhead.

Option C (SageMaker XGBoost training job with Lambda inference): While using SageMaker built-in XGBoost is a good starting point, this option still requires manual feature engineering, hyperparameter tuning, and deployment logic to Lambda, increasing the overall effort compared to Autopilot. It also requires the data scientists to choose XGBoost beforehand, potentially missing out on more accurate algorithms.
In summary, SageMaker Autopilot provides the quickest path to improve model performance with the least operational overhead, making it the most appropriate solution.

Question: 181

Exam Heist

A data scientist is reviewing customer comments about a company's products. The data scientist needs to present an initial exploratory analysis by using charts and a word cloud. The data scientist must use feature engineering techniques to prepare this analysis before starting a natural language processing (NLP) model.
Which combination of feature engineering techniques should the data scientist use to meet these requirements? (Choose two.)

- [ ] Named entity recognition
- [ ] Coreference
- [x] Stemming ✓
- [x] Term frequency-inverse document frequency (TF-IDF) ✓
- [ ] Sentiment analysis

Explanation:
The correct answer is C and D. Here's why:
C. Stemming: Stemming is a text normalization technique that reduces words to their root form. This is crucial for initial exploratory analysis and word clouds because it groups variations of the same word (e.g., "running," "runs," "ran" become "run"), consolidating frequencies and providing a more accurate representation of prominent themes in the customer comments. For example, if both "running" and "runs" are frequently mentioned, stemming will aggregate these instances under "run," making the analysis more meaningful.
D. Term Frequency-Inverse Document Frequency (TF-IDF): TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. It aims to highlight the most relevant words within the customer comments by considering both the frequency of a term within a single comment (TF) and its rarity across all comments (IDF). This technique helps to identify the most distinctive and important terms, making the word cloud and initial charts more informative. TF-IDF assigns higher weights to terms that appear frequently in a single comment but rarely across all comments, highlighting the most significant aspects of individual customer experiences. For example, the word "battery" might be common, but "battery life" coupled with a high sentiment score would be weighted higher to indicate the sentiment trend.
Why the other options are less suitable:

A. Named entity recognition (NER): NER identifies and classifies named entities in text (e.g., people, organizations, locations). While valuable for some NLP tasks, it's not essential for a basic exploratory analysis using charts and word clouds focused on general themes.
B. Coreference: Coreference resolution identifies expressions that refer to the same entity in text (e.g., "John" and "he"). Again, it's beneficial for more complex NLP tasks, but not crucial for creating word clouds or simple charts for initial exploration.
E. Sentiment analysis: Sentiment analysis identifies the emotional tone expressed in the text. While beneficial to understanding opinions, sentiment is not a feature engineering technique for preparing the text for a word cloud. Feature engineering techniques are for preparing the *input* of an NLP model, and not for deriving output (like sentiment).
In summary, Stemming normalizes the text by reducing words to their root forms, and TF-IDF weights terms based on their frequency and rarity, making them both essential techniques for preparing text data for exploratory analysis and word cloud generation within AWS's ML environment.
Supporting Links:

Stemming: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
TF-IDF: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Question: 182

Exam Heist

A data scientist is evaluating a GluonTS on Amazon SageMaker DeepAR model. The evaluation metrics on the test set indicate that the coverage score is 0.489 and 0.889 at the 0.5 and 0.9 quantiles, respectively.
What can the data scientist reasonably conclude about the distributional forecast related to the test set?

- The coverage scores indicate that the distributional forecast is poorly calibrated. These scores should be approximately equal to each other at all quantiles.
- The coverage scores indicate that the distributional forecast is poorly calibrated. These scores should peak at the median and be lower at the tails.
- The coverage scores indicate that the distributional forecast is correctly calibrated. These scores should always fall below the quantile itself.
- The coverage scores indicate that the distributional forecast is correctly calibrated. These scores should be approximately equal to the quantile itself. ✓

Explanation:

Here's a detailed justification for why option D is the correct answer:

The question relates to evaluating the calibration of a probabilistic forecasting model, specifically a DeepAR model in GluonTS on Amazon SageMaker, using coverage scores. Coverage scores are a crucial metric when assessing the reliability of a model's uncertainty estimates. Ideally, a well-calibrated probabilistic forecast will produce prediction intervals that contain the true observed values a proportion of the time equal to the stated confidence level.

The provided coverage scores (0.489 at the 0.5 quantile and 0.889 at the 0.9 quantile) represent the proportion of actual values falling within the predicted intervals defined by the 50th and 90th percentiles, respectively. In a perfectly calibrated model, the coverage score for a given quantile should closely match the quantile value itself.

Therefore, if the model is well-calibrated, we expect the coverage score at the 0.5 quantile (median) to be close to 0.5 (50%), and the coverage score at the 0.9 quantile to be close to 0.9 (90%). The given values, 0.489 and 0.889, are close enough to 0.5 and 0.9, respectively, indicating that the model's distributional forecast is reasonably well-calibrated.

Option A is incorrect because it implies that coverage scores should be equal at *all* quantiles. This is fundamentally wrong; higher quantiles should have higher coverage scores. Option B is incorrect because coverage scores should ideally match the quantile value, not peak at the median and decrease at the tails. Option C is incorrect because the coverage score should match the quantile, not fall *below* it. If the coverage score is lower than the quantile, it indicates underconfidence in the model's predictions.

In essence, good calibration implies the model accurately quantifies its uncertainty. If the model claims a 90% confidence interval, it should contain the true value about 90% of the time. DeepAR in GluonTS, when properly trained, aims to provide such well-calibrated probabilistic forecasts. Evaluating coverage is an important check on whether it has succeeded.

For further reading on probabilistic forecasting and evaluation metrics, explore:

GluonTS documentation: https://gluon-ts.mxnet.io/
Amazon SageMaker JumpStart documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-demos.html
Research papers on probabilistic time series forecasting: Search for papers on calibration and coverage in time series forecasting using keywords like "probabilistic forecasting calibration," "coverage probability," and "quantile regression."

Question: 183

Exam Heist

An energy company has wind turbines, weather stations, and solar panels that generate telemetry data. The company wants to perform predictive maintenance on these devices. The devices are in various locations and have unstable internet connectivity.

A team of data scientists is using the telemetry data to perform machine learning (ML) to conduct anomaly detection and predict maintenance before the devices start to deteriorate. The team needs a scalable, secure, high-velocity data ingestion mechanism. The team has decided to use Amazon S3 as the data storage location.

Which approach meets these requirements?

- Ingest the data by using an HTTP API call to a web server that is hosted on Amazon EC2. Set up EC2 instances in an Auto Scaling configuration behind an Elastic Load Balancer to load the data into Amazon S3.
- Ingest the data over Message Queuing Telemetry Transport (MQTT) to AWS IoT Core. Set up a rule in AWS IoT Core to use Amazon Kinesis Data Firehose to send data to an Amazon Kinesis data stream that is configured to write to an S3 bucket.
- Ingest the data over Message Queuing Telemetry Transport (MQTT) to AWS IoT Core. Set up a rule in AWS IoT Core to direct all MQTT data to an Amazon Kinesis Data Firehose delivery stream that is configured to write to an S3 bucket. ✓
- Ingest the data over Message Queuing Telemetry Transport (MQTT) to Amazon Kinesis data stream that is configured to write to an S3 bucket.

Explanation:

The correct answer is C because it provides a scalable, secure, and high-velocity data ingestion mechanism suitable for devices with unstable internet connectivity. Let's break down why:

MQTT to AWS IoT Core: MQTT is a lightweight messaging protocol ideal for IoT devices with limited bandwidth and intermittent connectivity. AWS IoT Core provides a managed platform for device connectivity, security, and data management. https://aws.amazon.com/iot-core/

AWS IoT Core Rule: AWS IoT Core Rules Engine allows you to define actions based on MQTT messages. This enables routing data to other AWS services. https://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html
Amazon Kinesis Data Firehose: Kinesis Data Firehose is designed for reliable, real-time data streaming to destinations like Amazon S3. It automatically scales to handle high-velocity data ingestion and offers data transformation and compression options. https://aws.amazon.com/kinesis/data-firehose/
S3 Bucket: Amazon S3 provides scalable and durable object storage, ideal for storing the ingested telemetry data for analysis. https://aws.amazon.com/s3/
Why other options are less suitable:

A: While EC2 with Auto Scaling and ELB can handle HTTP traffic, it's a more complex and potentially less cost-effective solution for IoT data ingestion compared to AWS IoT Core. Managing and maintaining EC2 instances adds overhead.
B: Introducing an unnecessary Kinesis Data Stream between Firehose and S3 adds complexity and cost without significant benefit. Kinesis Data Firehose can directly deliver to S3.
D: Directly sending data from IoT devices to Kinesis Data Streams is possible but bypasses the device management, security, and protocol translation features of AWS IoT Core. IoT Core is specifically designed for managing and securing IoT device connections.

Question: 184

Exam Heist

A retail company collects customer comments about its products from social media, the company website, and customer call logs. A team of data scientists and engineers wants to find common topics and determine which products the customers are referring to in their comments. The team is using natural language processing (NLP) to build a model to help with this classification.
Each product can be classified into multiple categories that the company defines. These categories are related but are not mutually exclusive. For example, if there is mention of "Sample Yogurt" in the document of customer comments, then "Sample Yogurt" should be classified as "yogurt," "snack," and "dairy product."
The team is using Amazon Comprehend to train the model and must complete the project as soon as possible.
Which functionality of Amazon Comprehend should the team use to meet these requirements?

- Custom classification with multi-class mode
- Custom classification with multi-label mode ✓
- Custom entity recognition
- Built-in models

Explanation:
The correct answer is **B. Custom classification with multi-label mode.**
Here's why:

Problem Type: The scenario requires classifying customer comments into multiple categories for each product. This is a multi-label classification problem, as a single comment can belong to several categories (e.g., "yogurt," "snack," and "dairy product").
Amazon Comprehend's Capabilities: Amazon Comprehend offers custom classification, which allows you to train models for specific classification tasks. It supports two classification modes: multi-class and multi-label.
Multi-Class vs. Multi-Label:

Multi-class classification: Each document can only be assigned to one category. This is not suitable for this scenario because a customer comment may refer to multiple product categories simultaneously.
Multi-label classification: Each document can be assigned to multiple categories. This aligns perfectly with the requirement that a product can be classified into multiple categories.
Why other options are incorrect:

A. Custom classification with multi-class mode: This is incorrect as each comment can be assigned to multiple categories, not just one.
C. Custom entity recognition: Entity recognition identifies and categorizes specific entities (like people, places, organizations). While helpful, it doesn't directly address the need to classify entire documents into predefined product categories.
D. Built-in models: Built-in models may not be granular enough to handle custom product categories specific to the retail company. Customer classification allows for more specific and accurate categorizations.
In conclusion, custom classification with multi-label mode in Amazon Comprehend is the most appropriate solution for the given scenario because it allows the team to train a model that can classify each customer comment into multiple product categories, meeting the project requirements efficiently.
Authoritative Links:

Amazon Comprehend Custom Classification

Understanding Multi-Label Classification

Question: 185

Exam Heist

A data engineer is using AWS Glue to create optimized, secure datasets in Amazon S3. The data science team wants the ability to access the ETL scripts directly from Amazon SageMaker notebooks within a VPC. After this setup is complete, the data science team wants the ability to run the AWS Glue job and invoke the
SageMaker training job.
Which combination of steps should the data engineer take to meet these requirements? (Choose three.)

- [ ] Create a SageMaker development endpoint in the data science team's VPC.
- [x] Create an AWS Glue development endpoint in the data science team's VPC. ✓
- [x] Create SageMaker notebooks by using the AWS Glue development endpoint. ✓
- [ ] Create SageMaker notebooks by using the SageMaker console.
- [ ] Attach a decryption policy to the SageMaker notebooks.
- [x] Create an IAM policy and an IAM role for the SageMaker notebooks. ✓

Explanation:
The correct answer is BCF because it addresses the requirements of accessing Glue ETL scripts from SageMaker notebooks within a VPC and then running Glue jobs and invoking SageMaker training jobs.

B. Create an AWS Glue development endpoint in the data science team's VPC: A Glue development endpoint allows users to interactively develop, test, and debug Glue ETL scripts. By placing the endpoint within the VPC, the data science team can securely access and modify the ETL scripts directly from within their VPC environment. This step enables a secure connection between the SageMaker notebooks and the Glue ETL scripts. [AWS Glue Development Endpoint: https://docs.aws.amazon.com/glue/latest/dg/dev-endpoint.html]
C. Create SageMaker notebooks by using the AWS Glue development endpoint: Connecting the SageMaker notebooks to the Glue development endpoint is crucial. This connection allows the notebook to access and interact with the ETL scripts. The development endpoint acts as a gateway, enabling users to test and debug the ETL logic directly from the notebook environment, which is critical for iterative development and refinement.
F. Create an IAM policy and an IAM role for the SageMaker notebooks: An IAM role with an attached policy is essential to grant the SageMaker notebooks the necessary permissions to interact with other AWS services, specifically AWS Glue (to start jobs) and SageMaker (to start training jobs). The IAM policy should include permissions to execute Glue jobs and start SageMaker training jobs. Without these permissions, the notebooks will not be able to trigger the Glue and SageMaker processes, making the entire workflow ineffective. [SageMaker Roles: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html]
Option A is incorrect because a SageMaker development endpoint does not provide direct access to the Glue ETL scripts. Option D is incorrect because creating a notebook using the SageMaker console does not establish the necessary connectivity to AWS Glue resources within the VPC. Option E is incorrect because a decryption policy isn't directly related to enabling the notebook to execute Glue jobs or invoke SageMaker training jobs, although encryption best practices should be followed. The IAM role is how the required permissions are managed.

Question: 186

Exam Heist

A data engineer needs to provide a team of data scientists with the appropriate dataset to run machine learning training jobs. The data will be stored in Amazon S3. The data engineer is obtaining the data from an Amazon Redshift database and is using join queries to extract a single tabular dataset. A portion of the schema is as follows:

TransactionTimestamp (Timestamp)
CardName (Varchar)
CardNo (Varchar)

The data engineer must provide the data so that any row with a CardNo value of NULL is removed. Also, the TransactionTimestamp column must be separated into a TransactionDate column and a TransactionTime column. Finally, the CardName column must be renamed to NameOnCard.

The data will be extracted on a monthly basis and will be loaded into an S3 bucket. The solution must minimize the effort that is needed to set up infrastructure for the ingestion and transformation. The solution also must be automated and must minimize the load on the Amazon Redshift cluster.

Which solution meets these requirements?

- Set up an Amazon EMR cluster. Create an Apache Spark job to read the data from the Amazon Redshift cluster and transform the data. Load the data into the S3 bucket. Schedule the job to run monthly.
- Set up an Amazon EC2 instance with a SQL client tool, such as SQL Workbench/J, to query the data from the Amazon Redshift cluster directly Export the resulting dataset into a file. Upload the file into the S3 bucket. Perform these tasks monthly.
- Set up an AWS Glue job that has the Amazon Redshift cluster as the source and the S3 bucket as the destination. Use the built-in transforms Filter, Map, and RenameField to perform the required transformations. Schedule the job to run monthly. ✓
- Use Amazon Redshift Spectrum to run a query that writes the data directly to the S3 bucket. Create an AWS Lambda function to run the query monthly.

Explanation:
The best solution is C, using AWS Glue. Here's why:

Minimizes infrastructure setup: AWS Glue is a fully managed ETL (Extract, Transform, Load) service, eliminating the need to manage servers or clusters, unlike Amazon EMR (option A) or EC2 instances (option B).
Automated transformation: Glue provides built-in transforms like `Filter`, `Map`, and `RenameField` that can handle data cleansing and transformation requirements like removing NULL values, splitting the timestamp, and renaming columns, as per the question's instructions. This simplifies the process compared to writing custom code with Spark (option A) or manual exports (option B).
Redshift integration: Glue has native connectors to Amazon Redshift, enabling seamless data extraction.
Scheduling: Glue provides built-in scheduling capabilities, automating the monthly data extraction and transformation process.
Minimizes Redshift Load: Glue extracts data from Redshift, performs transformations in its environment, and then loads the output into S3, reducing the computational load on the Redshift cluster.
Redshift Spectrum Consideration: While Redshift Spectrum (option D) can query data and write to S3, it's primarily designed for querying data stored in S3 data lake using Redshift's SQL engine and it is an overkill if Redshift itself is the source. Also, a Lambda function to execute Redshift Spectrum query does not inherently provide transforms capabilities.
Therefore, AWS Glue offers a comprehensive, managed, and automated solution for extracting data from Amazon Redshift, transforming it, and loading it into Amazon S3 while minimizing infrastructure management and Redshift load.
Supporting Links:

AWS Glue: https://aws.amazon.com/glue/
AWS Glue Transforms: https://docs.aws.amazon.com/glue/latest/dg/transforms-built-in.html
Amazon Redshift: https://aws.amazon.com/redshift/

Question: 187

Exam Heist

A machine learning (ML) specialist wants to bring a custom training algorithm to Amazon SageMaker. The ML specialist implements the algorithm in a Docker container that is supported by SageMaker.

How should the ML specialist package the Docker container so that SageMaker can launch the training correctly?

- Specify the server argument in the ENTRYPOINT instruction in the Dockerfile.
- Specify the training program in the ENTRYPOINT instruction in the Dockerfile. ✓
- Include the path to the training data in the docker build command when packaging the container.
- Use a COPY instruction in the Dockerfile to copy the training program to the /opt/ml/train directory.

Explanation:
The correct answer is **B. Specify the training program in the ENTRYPOINT instruction in the Dockerfile.**
Here's a detailed justification:
Amazon SageMaker relies on the `ENTRYPOINT` instruction within the Dockerfile to understand how to initiate the training process. The `ENTRYPOINT` specifies the executable that will run when the container starts. For a custom training algorithm, the `ENTRYPOINT` should point to the script or program that contains the training logic. This allows SageMaker to automatically start the training job when the container is launched. SageMaker training jobs need a way to execute your code; the `ENTRYPOINT` is this mechanism. Without specifying the training program in the `ENTRYPOINT`, SageMaker wouldn't know which script to run to kick off the training.
Option A is incorrect because while a server might be involved in certain advanced scenarios, the primary requirement is to launch the training program itself, not necessarily a server. The training program might be single-threaded and doesn't require a server.Option C is incorrect. Training data is provided to the container through SageMaker configurations and isn't baked into the image itself during the docker build process. The location of the training data is typically passed as an argument to the

training script.Option D is incorrect because while copying the training program to `/opt/ml/train` is a good practice for organization (SageMaker provides this convention), it doesn't automatically execute the training program. You still need to tell SageMaker *how* to execute it, and that's done through the `ENTRYPOINT`.
Therefore, specifying the training program in the `ENTRYPOINT` instruction is crucial for SageMaker to initiate the training process correctly when launching the Docker container.
Further Research:

SageMaker Documentation on Using Docker Containers: https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo.html
SageMaker Example Notebooks (look for Docker examples): https://github.com/aws/amazon-sagemaker-examples
Dockerfile Reference: https://docs.docker.com/engine/reference/builder/

Question: 188

Exam Heist

An ecommerce company wants to use machine learning (ML) to monitor fraudulent transactions on its website. The company is using Amazon SageMaker to research, train, deploy, and monitor the ML models.

The historical transactions data is in a .csv file that is stored in Amazon S3. The data contains features such as the user's IP address, navigation time, average time on each page, and the number of clicks for each session. There is no label in the data to indicate if a transaction is anomalous.

Which models should the company use in combination to detect anomalous transactions? (Choose two.)

- [x] IP Insights ✓
- [ ] K-nearest neighbors (k-NN)
- [ ] Linear learner with a logistic function
- [x] Random Cut Forest (RCF) ✓
- [ ] XGBoost

Explanation:
Here's a detailed justification for why options A (IP Insights) and D (Random Cut Forest - RCF) are the most suitable combination for detecting anomalous transactions in this scenario:
The problem describes an anomaly detection scenario where the data is unlabeled, meaning we don't have examples of fraudulent vs. non-fraudulent transactions to learn from directly. This immediately points us towards unsupervised learning techniques.

IP Insights: This algorithm is designed specifically for identifying anomalous IP addresses based on their usage patterns. It leverages embeddings to represent IP addresses and learns relationships between them. Anomalous IP addresses, which might be used for fraudulent activity, would stand out in this learned space. This aligns perfectly with the problem context where user IP address is a given feature. https://docs.aws.amazon.com/sagemaker/latest/dg/ip-insights.html
Random Cut Forest (RCF): RCF is another unsupervised algorithm well-suited for anomaly detection. It works by randomly cutting the data space and measuring the average path length from a data point to the root of a tree. Anomalies generally require shorter paths to isolate, as they are different from the general distribution of data. RCF can be used to identify anomalous transactions based on the combination of other features like navigation time, average time on each page, and the number of clicks. https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html
Now, let's consider why the other options are less appropriate:

K-Nearest Neighbors (k-NN): While k-NN can be used for anomaly detection, it typically requires defining a distance threshold. Without labeled data, determining an appropriate threshold can be difficult. Also, k-NN can be computationally expensive for large datasets.
Linear learner with a logistic function: This is a supervised classification algorithm. It requires labeled data (fraudulent vs. non-fraudulent transactions) to train a model, which is not available in this scenario.
XGBoost: XGBoost is also a supervised learning algorithm that needs labeled data for training. It's powerful for classification and regression but unsuitable for unsupervised anomaly detection.
In summary, combining IP Insights and RCF allows the ecommerce company to leverage the specific characteristics of IP addresses for anomaly detection (IP Insights) while also considering the broader behavioral features of transactions (RCF) in an unsupervised manner.

Question: 189

Exam Heist

A healthcare company is using an Amazon SageMaker notebook instance to develop machine learning (ML) models. The company's data scientists will need to be able to access datasets stored in Amazon S3 to train the models. Due to regulatory requirements, access to the data from instances and services used for training must not be transmitted over the internet.

Which combination of steps should an ML specialist take to provide this access? (Choose two.)

- [x] Configure the SageMaker notebook instance to be launched with a VPC attached and internet access disabled. ✓
- [ ] Create and configure a VPN tunnel between SageMaker and Amazon S3.
- [x] Create and configure an S3 VPC endpoint Attach it to the VPC. ✓
- [ ] Create an S3 bucket policy that allows traffic from the VPC and denies traffic from the internet.
- [ ] Deploy AWS Transit Gateway Attach the S3 bucket and the SageMaker instance to the gateway.

Explanation:
The correct answer is AC. Here's why:

A. Configure the SageMaker notebook instance to be launched with a VPC attached and internet access disabled: Launching the SageMaker notebook instance within a VPC ensures that all traffic to and from the instance stays within the AWS network. Disabling internet access prevents any accidental or malicious data exfiltration over the public internet, adhering to the regulatory requirements. https://docs.aws.amazon.com/sagemaker/latest/dg/security-vpc.html
C. Create and configure an S3 VPC endpoint. Attach it to the VPC: An S3 VPC endpoint provides a private connection between the VPC and S3, without requiring traffic to traverse the internet. This ensures that the SageMaker notebook instance, residing in the VPC, can securely access the S3 datasets. https://docs.aws.amazon.com/vpc/latest/privatelink/vpc-endpoints-s3.html

By using a gateway endpoint for S3, the traffic is routed privately without the need for NAT gateways or internet gateways in your VPC for S3 access. This helps in maintaining data privacy and fulfilling regulatory compliance.
Let's examine why the other options are not the best choices:

B. Create and configure a VPN tunnel between SageMaker and Amazon S3: While a VPN provides secure communication, it is an unnecessarily complex solution for traffic within AWS. VPC endpoints offer a simpler and more direct private connection.
D. Create an S3 bucket policy that allows traffic from the VPC and denies traffic from the internet: While an S3 bucket policy that allows traffic from the VPC and denies from the internet is an essential security measure. It is usually used in conjunction with VPC endpoints to restrict bucket access. However, this option alone does not create the necessary private connectivity.
E. Deploy AWS Transit Gateway. Attach the S3 bucket and the SageMaker instance to the gateway: AWS Transit Gateway is a hub for connecting multiple VPCs and on-premises networks. While useful in complex network setups, it's overkill for a single VPC needing access to S3, which is better handled by a VPC endpoint for S3.

Question: 190

Exam Heist

A machine learning (ML) specialist at a retail company is forecasting sales for one of the company's stores. The ML specialist is using data from the past 10 years. The company has provided a dataset that includes the total amount of money in sales each day for the store. Approximately 5% of the days are missing sales data.

The ML specialist builds a simple forecasting model with the dataset and discovers that the model performs poorly. The performance is poor around the time of seasonal events, when the model consistently predicts sales figures that are too low or too high.

Which actions should the ML specialist take to try to improve the model's performance? (Choose two.)

- [ ] Add information about the store's sales periods to the dataset.
- [x] Aggregate sales figures from stores in the same proximity. ✓
- [x] Apply smoothing to correct for seasonal variation. ✓
- [ ] Change the forecast frequency from daily to weekly.
- [ ] Replace missing values in the dataset by using linear interpolation.

Explanation:
Let's analyze the provided options and justify why B and C are the most appropriate actions to improve the model's performance.
B. Aggregate sales figures from stores in the same proximity.

Aggregating sales figures from nearby stores can help to reduce noise and smooth out anomalies in the data. Individual store sales can be affected by local events or temporary issues that don't reflect the overall trend. By combining data from multiple stores in a similar area, the model can learn a more robust and generalizable pattern, which is especially useful given 5% missing data points in the target store. This is an application of ensemble methods, where the aggregated data simulates an averaging effect which reduces variance. Consider this a basic form of data augmentation, as you are synthetically enlarging the training dataset.

C. Apply smoothing to correct for seasonal variation.

The problem description explicitly mentions poor performance around seasonal events. Smoothing techniques, such as moving averages or exponential smoothing, can help to reduce the impact of these seasonal variations and make the underlying trend clearer. These techniques help to de-noise the time series data, highlighting the general trend. Without applying smoothing, these variations skew the predictions away from the actual values.

Why other options are less suitable:

A. Add information about the store's sales periods to the dataset: While potentially helpful, this option primarily focuses on *known* sales periods. The problem statement already mentions the seasonal events, which are likely pre-existing and well-documented. This option might not solve the issue if there are finer-grained sales period details needed.

D. Change the forecast frequency from daily to weekly: This will lead to data loss and the inability to identify patterns at a granular level. Though it mitigates daily fluctuations, it fails to tackle seasonal trends.

E. Replace missing values in the dataset by using linear interpolation: While linear interpolation can address the missing data issue, it does not directly address the seasonal forecasting errors. Linear interpolation only fills the gaps in the data, whereas the question is more specific about error at seasonal events, which interpolation isn't specifically designed to solve.

Supporting Links:

Time Series Analysis (Smoothing): https://otexts.com/fpp2/
Ensemble Learning: https://scikit-learn.org/stable/modules/ensemble.html

Question: 191

Exam Heist

A newspaper publisher has a table of customer data that consists of several numerical and categorical features, such as age and education history, as well as subscription status. The company wants to build a targeted marketing model for predicting the subscription status based on the table data.

Which Amazon SageMaker built-in algorithm should be used to model the targeted marketing?

- ○ Random Cut Forest (RCF)
- ○ XGBoost ✓
- ○ Neural Topic Model (NTM)
- ○ DeepAR forecasting

Explanation:

The correct answer is B, XGBoost. Here's why:

The problem describes a classification task: predicting subscription status (likely a binary outcome: subscribed or not subscribed) based on a mix of numerical and categorical features. Several Amazon SageMaker built-in algorithms exist, but XGBoost is a highly effective gradient boosting algorithm specifically designed for classification and regression problems. It excels in handling tabular data with both numerical and categorical features, and is known for its accuracy and robustness.

A. Random Cut Forest (RCF) is an anomaly detection algorithm, not suitable for supervised classification problems like predicting subscription status. RCF identifies data points that deviate significantly from the norm and is designed for unsupervised learning.https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html

C. Neural Topic Model (NTM) is used for topic modeling in text documents. It's completely irrelevant to the described tabular data and the classification problem. NTM uncovers latent semantic structures in text, making it unsuitable for predicting subscription status based on customer data.https://docs.aws.amazon.com/sagemaker/latest/dg/ntm.html

D. DeepAR forecasting is a time-series forecasting algorithm. It is designed to predict future values based on past sequences of data. While subscription data *could* be analyzed over time, the question emphasizes predicting status based on customer characteristics (age, education) at a single point in time, not a time series. Therefore, DeepAR is not appropriate here.https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html

XGBoost, on the other hand, is well-suited for classification tasks with mixed feature types. Its strengths include:

Handling mixed data types: XGBoost natively handles both numerical and categorical features.
Regularization techniques: XGBoost incorporates L1 and L2 regularization to prevent overfitting, which is critical when dealing with complex datasets.

Gradient boosting: XGBoost is a gradient boosting algorithm, which sequentially builds an ensemble of decision trees, each correcting the errors of its predecessors, leading to high accuracy.
Scalability: XGBoost is designed to be scalable and can handle large datasets efficiently.
Therefore, XGBoost is the most appropriate Amazon SageMaker built-in algorithm for the newspaper publisher's targeted marketing model.https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html

Question: 192

Exam Heist

A company will use Amazon SageMaker to train and host a machine learning model for a marketing campaign. The data must be encrypted at rest. Most of the data is sensitive customer data. The company wants AWS to maintain the root of trust for the encryption keys and wants key usage to be logged.

Which solution will meet these requirements with the LEAST operational overhead?

- Use AWS Security Token Service (AWS STS) to create temporary tokens to encrypt the storage volumes for all SageMaker instances and to encrypt the model artifacts and data in Amazon S3.
- Use customer managed keys in AWS Key Management Service (AWS KMS) to encrypt the storage volumes for all SageMaker instances and to encrypt the model artifacts and data in Amazon S3. ✓
- Use encryption keys stored in AWS CloudHSM to encrypt the storage volumes for all SageMaker instances and to encrypt the model artifacts and data in Amazon S3.
- Use SageMaker built-in transient keys to encrypt the storage volumes for all SageMaker instances. Enable default encryption ffnew Amazon Elastic Block Store (Amazon EBS) volumes.

Explanation:
The correct answer is B, using customer-managed keys in AWS KMS. This solution effectively addresses the company's encryption at rest requirement for sensitive customer data used in SageMaker.
Here's why:

Encryption at Rest: AWS KMS allows encryption of both the SageMaker storage volumes (used by instances for training) and the model artifacts and data stored in Amazon S3. SageMaker integrates seamlessly with KMS for this purpose.
AWS Managed Root of Trust: KMS is a managed service, meaning AWS handles the underlying hardware and operational aspects of the key management infrastructure. The root of trust resides within AWS's secure infrastructure.
Key Usage Logging: KMS automatically logs all usage of the customer-managed keys in AWS CloudTrail. This provides a detailed audit trail of who accessed the keys and when, ensuring compliance and security monitoring.
Least Operational Overhead: Using KMS requires minimal operational overhead compared to other options. AWS manages the key lifecycle, rotation, and underlying HSMs (Hardware Security Modules). You simply manage the keys themselves, set appropriate permissions, and configure SageMaker to use them.
Let's analyze why the other options are less suitable:

A. AWS STS: STS generates temporary credentials for accessing AWS resources, but it doesn't directly encrypt data at rest. While STS credentials can be used to *access* encrypted data, STS doesn't *provide* the encryption itself or manage encryption keys.
C. AWS CloudHSM: CloudHSM provides the highest level of control over cryptographic keys, but it comes with significantly increased operational overhead. The customer is responsible for managing the HSM cluster, including patching, maintenance, and high availability. The requirement specifically mentioned *least* operational overhead, making CloudHSM unsuitable.
D. SageMaker Built-in Transient Keys and EBS Default Encryption: SageMaker's built-in transient keys are suitable for encrypting data in transit during training jobs, but not for encryption at rest. Further, while default EBS encryption encrypts EBS volumes, it leverages AWS-managed keys, which do not fulfill the requirement of AWS maintaining the root of trust while allowing the *company* to manage the key. The company wants to retain control and auditability using keys that they manage.
In summary, using customer-managed keys in AWS KMS strikes the best balance between security, control, and operational efficiency, perfectly aligning with the company's requirements.
Authoritative Links:

AWS KMS: https://aws.amazon.com/kms/
SageMaker Data Encryption: https://docs.aws.amazon.com/sagemaker/latest/dg/encryption-at-rest.html
AWS CloudTrail: https://aws.amazon.com/cloudtrail/

Question: 193

Exam Heist

A data scientist is working on a model to predict a company's required inventory stock levels. All historical data is stored in .csv files in the company's data lake on Amazon S3. The dataset consists of approximately 500 GB of data The data scientist wants to use SQL to explore the data before training the model. The company wants to minimize costs.

Which option meets these requirements with the LEAST operational overhead?

- Create an Amazon EMR cluster. Create external tables in the Apache Hive metastore, referencing the data that is stored in the S3 bucket. Explore the data from the Hive console.
- Use AWS Glue to crawl the S3 bucket and create tables in the AWS Glue Data Catalog. Use Amazon Athena to explore the data. ✓
- Create an Amazon Redshift cluster. Use the COPY command to ingest the data from Amazon S3. Explore the data from the Amazon Redshift query editor GUI.
- Create an Amazon Redshift cluster. Create external tables in an external schema, referencing the S3 bucket that contains the data. Explore the data from the Amazon Redshift query editor GUI.

Explanation:
Option B, using AWS Glue and Amazon Athena, is the most suitable solution for exploring the 500 GB dataset stored in S3 with SQL while minimizing costs and operational overhead. AWS Glue automatically discovers the schema of the CSV files through its crawler and populates the AWS Glue Data Catalog. This eliminates the manual effort of defining table schemas, crucial for a large dataset. Athena then allows querying the data directly from S3 using standard SQL.
Athena is serverless, so you only pay for the queries you run. This makes it a cost-effective solution for exploratory data analysis, especially when compared to maintaining a constantly running cluster like Amazon EMR (Option A) or Amazon Redshift (Options C and D). EMR involves cluster management and is overkill for simple SQL-based exploration.
While Amazon Redshift can query data in S3 using external tables, setting up and maintaining a Redshift cluster incurs higher costs compared to Athena's pay-per-query model. Ingesting the data into Redshift (Option C) adds significant time and cost.
Option D, while using external tables, still involves the overhead of managing the Redshift cluster, making it less efficient than Athena for initial data exploration.
Therefore, Glue and Athena provide a serverless, cost-effective, and low-operational-overhead approach to exploring the data in S3 using SQL, perfectly aligning with the problem's requirements.
Relevant links for further research:

AWS Glue: https://aws.amazon.com/glue/
Amazon Athena: https://aws.amazon.com/athena/
Amazon Redshift Spectrum (External Tables): https://docs.aws.amazon.com/redshift/latest/dg/c-using-spectrum.html

Question: 194

Exam Heist

A geospatial analysis company processes thousands of new satellite images each day to produce vessel detection data for commercial shipping. The company stores the training data in Amazon S3. The training data incrementally increases in size with new images each day.

The company has configured an Amazon SageMaker training job to use a single ml.p2.xlarge instance with File input mode to train the built-in Object Detection algorithm. The training process was successful last month but is now failing because of a lack of storage. Aside from the addition of training data, nothing has changed in the model training process.

A machine learning (ML) specialist needs to change the training configuration to fix the problem. The solution must optimize performance and must minimize the cost of training.

Which solution will meet these requirements?

- Modify the training configuration to use two ml.p2.xlarge instances.
- Modify the training configuration to use Pipe input mode. ✓
- Modify the training configuration to use a single ml.p3.2xlarge instance.
- Modify the training configuration to use Amazon Elastic File System (Amazon EFS) instead of Amazon S3 to store the input training data.

Explanation:
The optimal solution is **B. Modify the training configuration to use Pipe input mode.**
Here's why:

Problem: The training job is failing due to a lack of storage because the increasing training dataset size exceeds the local disk capacity of the `ml.p2.xlarge` instance when using File input mode. File mode downloads the entire dataset to the instance's local storage before training begins.

Why Pipe Mode is Better: Pipe input mode streams data directly from Amazon S3 to the training instance. This means the entire dataset doesn't need to be stored on the instance's local disk, resolving the storage limitation. It allows the algorithm to read data as it's being streamed, which makes it efficient for larger datasets.
Why other options are less suitable:

A. Modify the training configuration to use two ml.p2.xlarge instances: Increasing the number of instances does not address the root cause of the storage limitation on a *single* instance. It also increases cost, which is against the requirements.
C. Modify the training configuration to use a single ml.p3.2xlarge instance: While a `ml.p3.2xlarge` instance has more local storage than `ml.p2.xlarge`, it is still limited. Plus, simply increasing the instance size may not be cost-effective if the data keeps growing. Switching to the larger p3 instance type also doesn't address the fundamental problem that the instance's storage is still a bottleneck.
D. Modify the training configuration to use Amazon Elastic File System (Amazon EFS) instead of Amazon S3 to store the input training data: While EFS can provide more storage, accessing data from EFS can be slower and more expensive than accessing it directly from S3, especially when training deep learning models. The question highlights the performance must be optimized, this is an unnecessary change. S3 is built for handling large quantities of data effectively.
Cost and Performance: Pipe mode offers better performance for large datasets as it avoids the overhead of downloading and storing the entire dataset locally. In terms of cost, Pipe mode helps you avoid the cost of using larger instances or additional instances simply to accommodate the data size.
Authoritative Links:

SageMaker Input Data Configuration
Amazon SageMaker FAQs (search for "Pipe Mode")

Question: 195

Exam Heist

A company is using Amazon SageMaker to build a machine learning (ML) model to predict customer churn based on customer call transcripts. Audio files from customer calls are located in an on-premises VoIP system that has petabytes of recorded calls. The on-premises infrastructure has high-velocity networking and connects to the company's AWS infrastructure through a VPN connection over a 100 Mbps connection.

The company has an algorithm for transcribing customer calls that requires GPUs for inference. The company wants to store these transcriptions in an Amazon S3 bucket in the AWS Cloud for model development.

Which solution should an ML specialist use to deliver the transcriptions to the S3 bucket as quickly as possible?

- Order and use an AWS Snowball Edge Compute Optimized device with an NVIDIA Tesla module to run the transcription algorithm. Use AWS DataSync to send the resulting transcriptions to the transcription S3 bucket. ✓
- Order and use an AWS Snowcone device with Amazon EC2 Inf1 instances to run the transcription algorithm. Use AWS DataSync to send the resulting transcriptions to the transcription S3 bucket.
- Order and use AWS Outposts to run the transcription algorithm on GPU-based Amazon EC2 instances. Store the resulting transcriptions in the transcription S3 bucket.
- Use AWS DataSync to ingest the audio files to Amazon S3. Create an AWS Lambda function to run the transcription algorithm on the audio files when they are uploaded to Amazon S3. Configure the function to write the resulting transcriptions to the transcription S3 bucket.

Explanation:
Here's a detailed justification for why option A is the best solution, along with explanations of why the other options are less suitable, incorporating relevant cloud computing concepts and authoritative links.
Justification for Option A (AWS Snowball Edge Compute Optimized with DataSync):
The problem highlights a few key constraints: a large volume of on-premises audio data (petabytes), a slow network connection (100 Mbps), and the need for GPUs to run the transcription algorithm. Transferring petabytes of data over a 100 Mbps connection would take an extremely long time, making direct transfer impractical. Option A addresses these constraints effectively.
AWS Snowball Edge Compute Optimized is a suitable choice because it brings compute and storage capabilities to the edge, near the data source. The "Compute Optimized" variant, specifically with an NVIDIA Tesla module, provides the necessary GPU power to run the transcription algorithm locally on the Snowball Edge device. This eliminates the need to transfer the large audio files to the cloud for processing.https://aws.amazon.com/snowball/
After transcription, the resulting text files are significantly smaller than the original audio files. AWS DataSync is then used to efficiently transfer these smaller transcriptions to the designated S3 bucket. DataSync is optimized for transferring data over networks, offering features like incremental transfer, compression, and encryption, which maximizes the utilization of the 100 Mbps connection.https://aws.amazon.com/datasync/
The combination of local processing on Snowball Edge and efficient data transfer with DataSync provides the fastest way to deliver the transcriptions to S3 for model development, given the constraints.
Why Other Options Are Less Suitable:

Option B (AWS Snowcone): Snowcone is designed for smaller workloads and has limited compute resources, making it less ideal for computationally intensive GPU-based transcription. EC2 Inf1 instances are designed for inference but aren't compatible with Snowcone.https://aws.amazon.com/snowcone/
Option C (AWS Outposts): AWS Outposts brings AWS infrastructure to on-premises environments, but it's designed for persistent infrastructure needs. It's a more complex and expensive solution than Snowball Edge for a one-time data transfer and processing task. It also doesn't alleviate the bandwidth bottleneck.https://aws.amazon.com/outposts/
Option D (AWS DataSync to S3 and Lambda): While Lambda functions can process data in S3, transferring petabytes of audio files over a 100 Mbps connection is impractical. Furthermore, running the GPU-intensive transcription algorithm in Lambda would be difficult as Lambda's standard execution environments don't provide GPUs. Even if GPUs were available, running them on a Lambda function at scale would be prohibitively expensive. The initial bottleneck of data transfer remains unresolved.https://aws.amazon.com/lambda/
In summary: Option A provides the most efficient and cost-effective solution by processing the data locally on AWS Snowball Edge, leveraging its compute capabilities, and then using AWS DataSync to transfer the resulting transcriptions to S3 efficiently.

Question: 196

Exam Heist

A company has a podcast platform that has thousands of users. The company has implemented an anomaly detection algorithm to detect low podcast engagement based on a 10-minute running window of user events such as listening, pausing, and exiting the podcast. A machine learning (ML) specialist is designing the data ingestion of these events with the knowledge that the event payload needs some small transformations before inference.

How should the ML specialist design the data ingestion to meet these requirements with the LEAST operational overhead?

- Ingest event data by using a GraphQLAPI in AWS AppSync. Store the data in an Amazon DynamoDB table. Use DynamoDB Streams to call an AWS Lambda function to transform the most recent 10 minutes of data before inference.
- Ingest event data by using Amazon Kinesis Data Streams. Store the data in Amazon S3 by using Amazon Kinesis Data Firehose. Use AWS Glue to transform the most recent 10 minutes of data before inference.
- Ingest event data by using Amazon Kinesis Data Streams. Use an Amazon Kinesis Data Analytics for Apache Flink application to transform the most recent 10 minutes of data before inference. ✓
- Ingest event data by using Amazon Managed Streaming for Apache Kafka (Amazon MSK). Use an AWS Lambda function to transform the most recent 10 minutes of data before inference.

Explanation:
The correct answer is **C**. Here's why:

Real-time Anomaly Detection: The problem requires real-time anomaly detection on a running 10-minute window of user events. Kinesis Data Analytics for Apache Flink is specifically designed for real-time data processing and stream analytics. It allows for windowing operations (like the 10-minute window) and applying transformations directly on the streaming data as it arrives.
Data Transformation in Stream: Kinesis Data Analytics can perform the required small transformations on the event payload within the streaming application, before inference.
Low Operational Overhead: Kinesis Data Analytics is a managed service, meaning AWS handles the underlying infrastructure, scaling, and maintenance. This significantly reduces the operational overhead compared to managing your own Kafka cluster or writing custom code to manage data windows.
Why other options are less optimal:

A: AppSync and DynamoDB Streams introduce unnecessary complexity and latency. AppSync is designed for building APIs, not necessarily optimal for high-throughput data ingestion. DynamoDB Streams and Lambda are suitable for event-driven architectures, but not as efficient as Kinesis Data Analytics for this specific use case.
B: Kinesis Data Firehose is primarily used for delivering streaming data to destinations like S3. While AWS Glue can transform data, it is not the best choice for real-time, low-latency transformations needed for anomaly detection. Glue is more suitable for batch processing and ETL tasks. Storing the data in S3 introduces extra latency compared to real-time stream analysis.
D: Amazon MSK is a managed Kafka service, which is a good choice for high-throughput data ingestion. However, using a Lambda function for transformations within a 10-minute window adds complexity in window management and stateful processing. Kinesis Data Analytics simplifies this process with built-in windowing and state management capabilities.
In summary, Kinesis Data Analytics offers the most streamlined and efficient solution for ingesting, transforming, and analyzing the streaming event data in real time with minimal operational overhead.
Relevant Links:

Amazon Kinesis Data Analytics

Apache Flink

Question: 197

Exam Heist

A company wants to predict the classification of documents that are created from an application. New documents are saved to an Amazon S3 bucket every 3 seconds. The company has developed three versions of a machine learning (ML) model within Amazon SageMaker to classify document text. The company wants to deploy these three versions to predict the classification of each document.

Which approach will meet these requirements with the LEAST operational overhead?

- Configure an S3 event notification that invokes an AWS Lambda function when new documents are created. Configure the Lambda function to create three SageMaker batch transform jobs, one batch transform job for each model for each document.
- Deploy all the models to a single SageMaker endpoint. Treat each model as a production variant. Configure an S3 event notification that invokes an AWS Lambda function when new documents are created. Configure the Lambda function to call each production variant and return the results of each model. ✓
- Deploy each model to its own SageMaker endpoint Configure an S3 event notification that invokes an AWS Lambda function when new documents are created. Configure the Lambda function to call each endpoint and return the results of each model.
- Deploy each model to its own SageMaker endpoint. Create three AWS Lambda functions. Configure each Lambda function to call a different endpoint and return the results. Configure three S3 event notifications to invoke the Lambda functions when new documents are created.

Explanation:
The correct answer is B because it offers the most efficient and cost-effective solution with the least operational overhead for deploying and managing multiple ML models for real-time inference on documents uploaded to S3.
Option B leverages a SageMaker endpoint with multiple production variants. This allows all three models to be hosted on the same endpoint, sharing resources and reducing infrastructure costs. This contrasts with options C and D, where each model requires its own dedicated endpoint, leading to increased costs and management complexity. SageMaker endpoints are designed for real-time inference and can handle frequent requests.
[https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html]
An S3 event notification triggers a Lambda function upon document creation. This Lambda function then calls the SageMaker endpoint, specifying each production variant (model) within the request. This makes the deployment process less complex than deploying three dedicated Lambda functions as in option D. The Lambda function orchestrates the invocation of all three models.
Option A utilizes SageMaker Batch Transform, which is designed for processing large datasets offline, not for real-time inference triggered every 3 seconds. Batch transform is more suitable for tasks where latency is not a critical factor. Therefore, option A does not meet the real-time requirements of the scenario.
[https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html]
The Lambda function in Option B can handle the data transformation and forwarding, eliminating the need for multiple Lambda functions or batch transforms. All the models are efficiently managed within a single SageMaker endpoint. This is efficient and cost effective. It will use computing resource only when called. It is the most efficient. This approach minimizes latency and operational overhead and is the most streamlined way to serve real-time predictions from multiple models in SageMaker based on S3 events.

Question: 198

Exam Heist

A manufacturing company needs to identify returned smartphones that have been damaged by moisture. The company has an automated process that produces 2,000 diagnostic values for each phone. The database contains more than five million phone evaluations. The evaluation process is consistent, and there are no missing values in the data. A machine learning (ML) specialist has trained an Amazon SageMaker linear learner ML model to classify phones as moisture damaged or not moisture damaged by using all available features. The model's F1 score is 0.6.

Which changes in model training would MOST likely improve the model's F1 score? (Choose two.)

- [x] Continue to use the SageMaker linear learner algorithm. Reduce the number of features with the SageMaker principal component analysis (PCA) algorithm. ✓
- [ ] Continue to use the SageMaker linear learner algorithm. Reduce the number of features with the scikit-learn multi-dimensional scaling (MDS) algorithm.
- [ ] Continue to use the SageMaker linear learner algorithm. Set the predictor type to regressor.
- [ ] Use the SageMaker k-means algorithm with k of less than 1,000 to train the model.
- [x] Use the SageMaker k-nearest neighbors (k-NN) algorithm. Set a dimension reduction target of less than 1,000 to train the model. ✓

Explanation:
The problem indicates a low F1 score (0.6) for a linear learner model trained on high-dimensional data (2,000 features) to classify moisture damage. A low F1 score often signifies a combination of low precision and low recall, pointing to issues like overfitting or poor feature representation.
Option A suggests using Principal Component Analysis (PCA) with the linear learner. PCA is a dimensionality reduction technique that transforms the original features into a set of uncorrelated principal components, capturing most of the variance in the data with fewer dimensions. This addresses potential overfitting caused by the high number of features and improves model generalization, thereby potentially improving the F1 score. Using SageMaker's built-in PCA algorithm seamlessly integrates with the existing SageMaker workflow. [https://docs.aws.amazon.com/sagemaker/latest/dg/pca-howto.html]
Option E proposes using k-Nearest Neighbors (k-NN) with dimension reduction. k-NN is a non-parametric algorithm that can capture non-linear relationships in the data, which might be missed by a linear learner. High dimensionality can negatively impact k-NN performance (curse of dimensionality). Reducing the dimensions before applying k-NN alleviates this issue and enables k-NN to more effectively identify similar phones based on the reduced feature set. SageMaker's k-NN supports built-in dimension reduction. [https://docs.aws.amazon.com/sagemaker/latest/dg/knn-algorithm.html]
Option B uses Multi-Dimensional Scaling (MDS). While MDS is a dimensionality reduction technique, it's primarily for visualizing high-dimensional data in lower dimensions while preserving distances. It isn't typically used as a pre-processing step for classification tasks in the same way PCA is. PCA is often preferred for feature extraction in classification.
Option C suggests changing the predictor type to regressor. The problem is a classification problem (moisture damaged or not). Changing the predictor type to regressor would fundamentally change the model's objective, making it unsuitable for the task and likely degrading performance. Regression predicts a continuous value, not a class label.
Option D suggests using k-means. k-means is a clustering algorithm, used for grouping data points. While it can be used for anomaly detection, it's not directly suitable for classification without further processing (e.g., labeling clusters after they are formed). Furthermore, choosing k < 1000 is arbitrary and doesn't address the core issue of dimensionality reduction for classification. The manufacturing company already has labels, so the use of an unsupervised learning algorithm like k-means is less appropriate.

Question: 199

Exam Heist

A company is building a machine learning (ML) model to classify images of plants. An ML specialist has trained the model using the Amazon SageMaker built-in Image Classification algorithm. The model is hosted using a SageMaker endpoint on an ml.m5.xlarge instance for real-time inference. When used by researchers in the field, the inference has greater latency than is acceptable. The latency gets worse when multiple researchers perform inference at the same time on their devices. Using Amazon CloudWatch metrics, the ML specialist notices that the ModelLatency metric shows a high value and is responsible for most of the response latency.

The ML specialist needs to fix the performance issue so that researchers can experience less latency when performing inference from their devices.

Which action should the ML specialist take to meet this requirement?

- Change the endpoint instance to an ml.t3 burstable instance with the same vCPU number as the ml.m5.xlarge instance has.
- Attach an Amazon Elastic Inference ml.eia2.medium accelerator to the endpoint instance. ✓
- Enable Amazon SageMaker Autopilot to automatically tune performance of the model.
- Change the endpoint instance to use a memory optimized ML instance.

Explanation:
The correct answer is **B. Attach an Amazon Elastic Inference ml.eia2.medium accelerator to the endpoint instance.**
Here's why:

The Problem: The high `ModelLatency` in CloudWatch indicates that the model inference itself is taking a significant amount of time, causing overall response latency. This worsens with concurrent requests.
Why Elastic Inference (EI) is Suitable: Elastic Inference is designed to reduce the cost of running deep learning inference by allowing you to attach just the right amount of GPU-powered acceleration to your existing Amazon EC2 or SageMaker instances. In this case, the ml.m5.xlarge instance might be CPU-bound during inference, and adding a GPU accelerator can drastically reduce the inference time (`ModelLatency`). The `ml.eia2.medium` accelerator is a cost-effective option for adding GPU acceleration.
Why other options are not:

A: Using a `ml.t3` burstable instance isn't appropriate for consistent inference workloads. Burstable instances are designed for infrequent, short bursts of high CPU usage, and their performance can degrade significantly when sustained CPU usage is required, which is likely the case with real-time inference requests.
C: SageMaker Autopilot automates the model building process but does not continuously optimize the performance of a deployed endpoint in real-time. Autopilot is mostly focused on finding the best model type, and hyperparameters, it is not an

appropriate answer for tuning an already deployed model.
D: Memory optimization does not improve ModelLatency, Memory optimization is important if the model is too big for the instance type.
Justification: Attaching an Elastic Inference accelerator directly addresses the high `ModelLatency` by offloading the computationally intensive inference tasks from the CPU to the GPU. This reduces inference time, handles concurrent requests better, and can improve the overall throughput of the endpoint.
Supporting Links:

Amazon Elastic Inference Documentation
SageMaker Inference

Question: 200

Exam Heist

An automotive company is using computer vision in its autonomous cars. The company has trained its models successfully by using transfer learning from a convolutional neural network (CNN). The models are trained with PyTorch through the use of the Amazon SageMaker SDK. The company wants to reduce the time that is required for performing inferences, given the low latency that is required for self-driving.

Which solution should the company use to evaluate and improve the performance of the models?

- Use Amazon CloudWatch algorithm metrics for visibility into the SageMaker training weights, gradients, biases, and activation outputs. Compute the filter ranks based on this information. Apply pruning to remove the low-ranking filters. Set the new weights. Run a new training job with the pruned model.
- Use SageMaker Debugger for visibility into the training weights, gradients, biases, and activation outputs. Adjust the model hyperparameters, and look for lower inference times. Run a new training job.
- Use SageMaker Debugger for visibility into the training weights, gradients, biases, and activation outputs. Compute the filter ranks based on this information. Apply pruning to remove the low-ranking filters. Set the new weights. Run a new training job with the pruned model. ✓
- Use SageMaker Model Monitor for visibility into the ModelLatency metric and OverheadLatency metric of the model after the model is deployed. Adjust the model hyperparameters, and look for lower inference times. Run a new training job.

Explanation:
The correct answer is C because it provides a method for model optimization that directly addresses the need for reduced inference time in the context of autonomous driving. The process involves inspecting the model's internal state during training using SageMaker Debugger, identifying less important filters based on computed ranks, pruning these filters to simplify the model, and then retraining the pruned model to fine-tune it. This approach, known as model pruning, directly reduces the model's complexity, leading to faster inference times, which is crucial for the low-latency requirements of self-driving cars.
Option A is incorrect because CloudWatch algorithm metrics, while useful, don't provide the granular visibility into the training process that SageMaker Debugger offers. CloudWatch is more suited for monitoring the overall training process, not for in-depth analysis of model internals.
Option B focuses on hyperparameter tuning, which can impact inference time, but it doesn't directly address model size and complexity as pruning does. Hyperparameter tuning can be complementary to pruning but isn't a direct solution for reducing inference latency in the same way.
Option D utilizes SageMaker Model Monitor, which observes model performance after deployment, focusing on latency metrics. While valuable for detecting performance degradation and concept drift, Model Monitor doesn't offer mechanisms to improve the model architecture for reduced latency proactively before deployment. It is a reactive approach rather than a proactive optimization. Model Monitor can provide valuable feedback, it does not provide a means to optimise the model weights.
Therefore, the best solution is C because it directly addresses the problem of inference time reduction through model pruning, guided by detailed insights from SageMaker Debugger. This aligns with the objective of optimizing models for low-latency inference.
Relevant links for further research:

SageMaker Debugger: https://docs.aws.amazon.com/sagemaker/latest/dg/debugger.html
Model Pruning: https://docs.aws.amazon.com/sagemaker/latest/dg/model-optimization-pruning.html

Question: 201

Exam Heist

A company's machine learning (ML) specialist is designing a scalable data storage solution for Amazon SageMaker. The company has an existing TensorFlow-based model that uses a train.py script. The model relies on static training data that is currently stored in TFRecord format.

What should the ML specialist do to provide the training data to SageMaker with the LEAST development overhead?

- Put the TFRecord data into an Amazon S3 bucket. Use AWS Glue or AWS Lambda to reformat the data to protobuf format and store the data in a second S3 bucket. Point the SageMaker training invocation to the second S3 bucket.
- Rewrite the train.py script to add a section that converts TFRecord data to protobuf format. Point the SageMaker training invocation to the local path of the data. Ingest the protobuf data instead of the TFRecord data.
- Use SageMaker script mode, and use train.py unchanged. Point the SageMaker training invocation to the local path of the data without reformatting the training data.
- Use SageMaker script mode, and use train.py unchanged. Put the TFRecord data into an Amazon S3 bucket. Point the SageMaker training invocation to the S3 bucket without reformatting the training data. ✓

Explanation:
The correct answer is D because it offers the least development overhead while leveraging SageMaker's built-in capabilities for handling common data formats like TFRecord.
Here's a detailed justification:
SageMaker script mode allows you to bring your own training script (train.py in this case) and have SageMaker execute it. This eliminates the need to refactor existing code significantly. Since the model already works with TFRecord, there's no need to convert it to protobuf or any other format, saving development time and potential issues from introducing new code.
Amazon S3 is a scalable and cost-effective storage solution for large datasets. SageMaker can directly read data from S3 buckets, making it a seamless integration.
Option A introduces unnecessary complexity by requiring data reformatting with AWS Glue or Lambda. This adds development overhead, increases processing time, and potentially introduces errors during data transformation. Furthermore, it necessitates creating and managing additional AWS resources.
Option B involves modifying the existing train.py script to convert TFRecord to protobuf. This breaks the "least development overhead" requirement. Furthermore, the question implies the data volume is large, making the local path unsuitable for scalable training jobs. Ingesting protobuf data locally would be inefficient and likely infeasible for large training datasets.
Option C suggests using SageMaker script mode but uses the *local* path of the data. This is not a scalable solution. SageMaker training jobs typically run on multiple instances; using the local path would only provide the data to the instance where the training job was initiated. This is inadequate for distributed training. The question explicitly asks for a scalable solution.
Therefore, placing the existing TFRecord data in S3 and pointing SageMaker's training invocation to that S3 bucket directly utilizes SageMaker's built-in capabilities to handle the existing data format with the least amount of code modification or the introduction of additional services.
Relevant links for further research:

SageMaker Script Mode: https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo.html
SageMaker and S3: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-data-input-output.html
TFRecord: https://www.tensorflow.org/tutorials/load_data/tfrecord

Question: 202

Exam Heist

An ecommerce company wants to train a large image classification model with 10,000 classes. The company runs multiple model training iterations and needs to minimize operational overhead and cost. The company also needs to avoid loss of work and model retraining.

Which solution will meet these requirements?

- Create the training jobs as AWS Batch jobs that use Amazon EC2 Spot Instances in a managed compute environment.
- Use Amazon EC2 Spot Instances to run the training jobs. Use a Spot Instance interruption notice to save a snapshot of the model to Amazon S3 before an instance is terminated.
- Use AWS Lambda to run the training jobs. Save model weights to Amazon S3.
- Use managed spot training in Amazon SageMaker. Launch the training jobs with checkpointing enabled. ✓

Explanation:
The correct answer is D, using managed spot training in Amazon SageMaker with checkpointing enabled. Here's why:

Minimizing operational overhead: SageMaker managed spot training automates much of the infrastructure management, reducing the burden on the data science team. It automatically handles the provisioning and termination of Spot Instances, as well as retries in case of interruption. This aligns with the requirement of minimizing operational overhead.
Cost optimization: Spot Instances offer significant cost savings compared to On-Demand Instances. SageMaker's managed spot training leverages these savings without requiring manual management of instance bids and interruptions. This directly addresses the need to minimize cost.

Avoiding loss of work and retraining: Checkpointing saves the model's state periodically during training. In case of a Spot Instance interruption, SageMaker restarts the training job from the latest checkpoint. This prevents the loss of progress and avoids the need to retrain the model from scratch, satisfying the requirement to avoid work loss.
Large-scale image classification: SageMaker is designed to handle large datasets and complex models, making it suitable for training a model with 10,000 classes.
Why other options are less suitable:

A. AWS Batch with Spot Instances: While AWS Batch can manage compute resources, it requires more manual configuration and monitoring compared to SageMaker's managed spot training. Checkpointing would also need to be configured separately.
B. EC2 Spot Instances with interruption notice: This option requires manually handling Spot Instance interruptions and saving the model to S3. This increases operational overhead and the risk of errors.
C. AWS Lambda: Lambda functions have execution time limits and memory constraints that are not suitable for large-scale model training. Additionally, saving model weights to S3 would likely exceed Lambda's storage limitations.
Authoritative Links:

Amazon SageMaker Managed Spot Training: https://docs.aws.amazon.com/sagemaker/latest/dg/model-managed-spot-training.html
Amazon SageMaker Checkpointing: https://docs.aws.amazon.com/sagemaker/latest/dg/training-checkpoints.html
AWS Spot Instances: https://aws.amazon.com/ec2/spot/

Question: 203

Exam Heist

A retail company uses a machine learning (ML) model for daily sales forecasting. The model has provided inaccurate results for the past 3 weeks. At the end of each day, an AWS Glue job consolidates the input data that is used for the forecasting with the actual daily sales data and the predictions of the model. The AWS Glue job stores the data in Amazon S3.

The company's ML team determines that the inaccuracies are occurring because of a change in the value distributions of the model features. The ML team must implement a solution that will detect when this type of change occurs in the future.

Which solution will meet these requirements with the LEAST amount of operational overhead?

- Use Amazon SageMaker Model Monitor to create a data quality baseline. Confirm that the emit_metrics option is set to Enabled in the baseline constraints file. Set up an Amazon CloudWatch alarm for the metric. ✓
- Use Amazon SageMaker Model Monitor to create a model quality baseline. Confirm that the emit_metrics option is set to Enabled in the baseline constraints file. Set up an Amazon CloudWatch alarm for the metric.
- Use Amazon SageMaker Debugger to create rules to capture feature values Set up an Amazon CloudWatch alarm for the rules.
- Use Amazon CloudWatch to monitor Amazon SageMaker endpoints. Analyze logs in Amazon CloudWatch Logs to check for data drift.

Explanation:
The correct answer is A. Here's why:

Data Quality Monitoring vs. Model Quality Monitoring: The problem states that the *distribution of the model features* has changed. This directly points to a *data quality* issue, not a *model quality* issue (which would relate to the model's performance metrics like accuracy, precision, or recall). Therefore, we need to monitor the input data's characteristics. This eliminates option B.
Amazon SageMaker Model Monitor: SageMaker Model Monitor is specifically designed to monitor data quality, data drift, and model quality of ML models deployed in production. It can automatically create baselines for your data and detect deviations from those baselines.
Emit Metrics: Enabling the `emit_metrics` option in the baseline constraints file instructs Model Monitor to emit CloudWatch metrics related to the data quality checks. These metrics can include things like the mean, standard deviation, min, max, and number of missing values for each feature.
CloudWatch Alarms: Once the CloudWatch metrics are emitted, you can easily create CloudWatch alarms that trigger when these metrics exceed predefined thresholds. This provides an automated way to detect data drift or significant changes in feature distributions.
Operational Overhead: Option A has the least overhead because it leverages a managed service (SageMaker Model Monitor) with pre-built functionalities for data quality monitoring. The baseline creation and drift detection are automated, requiring minimal manual configuration or custom coding.
Why not C (SageMaker Debugger): SageMaker Debugger is mainly for debugging training jobs and analyzing model internal states during training. While it can capture feature values, it's not the primary tool for continuous data quality monitoring in production and has more overhead compared to Model Monitor for this specific task.

Why not D (CloudWatch Logs Analysis): Manually analyzing logs in CloudWatch Logs for data drift would require custom parsing logic, and scripting, adding significantly more operational overhead. This isn't a scalable or efficient solution for continuous data quality monitoring.
In summary, Amazon SageMaker Model Monitor with its data quality baseline, `emit_metrics` functionality, and CloudWatch alarms provides an automated and efficient solution for detecting changes in the distribution of model features with the least operational overhead.
References:

Amazon SageMaker Model Monitor
Monitor models in production

Question: 204

Exam Heist

A machine learning (ML) specialist has prepared and used a custom container image with Amazon SageMaker to train an image classification model. The ML specialist is performing hyperparameter optimization (HPO) with this custom container image to produce a higher quality image classifier.

The ML specialist needs to determine whether HPO with the SageMaker built-in image classification algorithm will produce a better model than the model produced by HPO with the custom container image. All ML experiments and HPO jobs must be invoked from scripts inside SageMaker Studio notebooks.

How can the ML specialist meet these requirements in the LEAST amount of time?

- Prepare a custom HPO script that runs multiple training jobs in SageMaker Studio in local mode to tune the model of the custom container image. Use the automatic model tuning capability of SageMaker with early stopping enabled to tune the model of the built-in image classification algorithm. Select the model with the best objective metric value.
- Use SageMaker Autopilot to tune the model of the custom container image. Use the automatic model tuning capability of SageMaker with early stopping enabled to tune the model of the built-in image classification algorithm. Compare the objective metric values of the resulting models of the SageMaker AutopilotAutoML job and the automatic model tuning job. Select the model with the best objective metric value.
- Use SageMaker Experiments to run and manage multiple training jobs and tune the model of the custom container image. Use the automatic model tuning capability of SageMaker to tune the model of the built-in image classification algorithm. Select the model with the best objective metric value. ✓
- Use the automatic model tuning capability of SageMaker to tune the models of the custom container image and the built-in image classification algorithm at the same time. Select the model with the best objective metric value.

Explanation:
The correct answer is C. Here's why:
Explanation
The question requires a comparison of HPO results between a custom container image and the SageMaker built-in image classification algorithm, all executed from SageMaker Studio notebooks with minimal time.

Option A (Incorrect): Running HPO in local mode within SageMaker Studio is not efficient for large-scale HPO. Local mode uses the resources of the notebook instance itself, which are limited. This would be slow and potentially not representative of performance in a production environment.
Option B (Incorrect): SageMaker Autopilot is designed for automated end-to-end machine learning pipelines, including feature engineering, algorithm selection, and hyperparameter optimization. While it could be used for the custom container, it introduces unnecessary overhead for the built-in algorithm when simply comparing HPO performance is needed. Also, using autopilot does not allow for running and managing multiple training jobs.
Option C (Correct): SageMaker Experiments is the ideal tool for organizing, tracking, and comparing multiple training runs, including HPO jobs. It allows for easy management and comparison of objective metrics across different experiments (custom container vs. built-in algorithm). Using automatic model tuning for the built-in algorithm is efficient, and SageMaker Experiments provides a framework for managing the multiple jobs for the custom container.
Option D (Incorrect): While the automatic model tuning capability of SageMaker can be used for both the custom container and the built-in algorithm, this option does not address the need for structured management and comparison of the experiments. SageMaker Experiments are not mentioned.
Justification
SageMaker Experiments provides the necessary features for running and tracking the multiple training jobs needed for HPO of the custom container. It allows for organizing the results of HPO for both the custom container and the built-in algorithm, enabling a direct comparison of the objective metric values. The automatic model tuning capability efficiently tunes the built-in algorithm, and SageMaker Experiments provides the structure to compare these results effectively. Therefore, option C is the most appropriate answer.
Supporting Links

SageMaker Experiments: https://docs.aws.amazon.com/sagemaker/latest/dg/experiments.html
SageMaker Automatic Model Tuning: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html

Question: 205

Exam Heist

A company wants to deliver digital car management services to its customers. The company plans to analyze data to predict the likelihood of users changing cars. The company has 10 TB of data that is stored in an Amazon Redshift cluster. The company's data engineering team is using Amazon SageMaker Studio for data analysis and model development. Only a subset of the data is relevant for developing the machine learning models. The data engineering team needs a secure and cost-effective way to export the data to a data repository in Amazon S3 for model development.

Which solutions will meet these requirements? (Choose two.)

- [ ] Launch multiple medium-sized instances in a distributed SageMaker Processing job. Use the prebuilt Docker images for Apache Spark to query and plot the relevant data and to export the relevant data from Amazon Redshift to Amazon S3.
- [ ] Launch multiple medium-sized notebook instances with a PySpark kernel in distributed mode. Download the data from Amazon Redshift to the notebook cluster. Query and plot the relevant data. Export the relevant data from the notebook cluster to Amazon S3.
- [x] Use AWS Secrets Manager to store the Amazon Redshift credentials. From a SageMaker Studio notebook, use the stored credentials to connect to Amazon Redshift with a Python adapter. Use the Python client to query the relevant data and to export the relevant data from Amazon Redshift to Amazon S3. ✓
- [ ] Use AWS Secrets Manager to store the Amazon Redshift credentials. Launch a SageMaker extra-large notebook instance with block storage that is slightly larger than 10 TB. Use the stored credentials to connect to Amazon Redshift with a Python adapter. Download, query, and plot the relevant data. Export the relevant data from the local notebook drive to Amazon S3.
- [x] Use SageMaker Data Wrangler to query and plot the relevant data and to export the relevant data from Amazon Redshift to Amazon S3. ✓

Explanation:
Let's analyze why options C and E are the most suitable solutions for exporting a subset of data from Amazon Redshift to Amazon S3 for model development in a secure and cost-effective manner using SageMaker Studio.
Option C: Using AWS Secrets Manager and a SageMaker Studio notebook with a Python adapter.
This approach emphasizes both security and practicality. Storing Redshift credentials in AWS Secrets Manager ensures that sensitive information is not hardcoded in the notebook, enhancing security. This aligns with best practices for managing secrets in cloud environments. From the SageMaker Studio notebook, a Python adapter (such as `psycopg2`) can connect to Redshift using the stored credentials. A SQL query can then be used to extract *only* the relevant data, avoiding the need to download the entire 10 TB dataset. This filtered data can then be efficiently exported to S3. SageMaker Studio provides a managed environment for data analysis and model development, making this a convenient choice. The approach is cost-effective as it only transfers the necessary data.
Option E: Using SageMaker Data Wrangler.
SageMaker Data Wrangler is specifically designed to simplify data preparation and feature engineering for machine learning. It offers a visual interface to connect to various data sources, including Amazon Redshift, and to perform transformations. It allows users to query, explore, clean, and transform data using a no-code or low-code interface. Data Wrangler can efficiently filter and sample the relevant data. The result can then be easily exported to Amazon S3. This reduces development time and effort compared to manual coding. It streamlines the process of preparing data for model training, and its integration with SageMaker Studio makes it a natural choice.
Why other options are less suitable:

Option A: While Spark on SageMaker Processing jobs is a viable option, it's more complex to set up than Data Wrangler or a simple Python script within a Studio notebook for this relatively straightforward task. Also, the question highlights SageMaker Studio as the tool of choice.
Option B: Using multiple notebook instances in distributed mode for PySpark is overkill for filtering a subset of data. It adds unnecessary complexity and cost compared to the other options.
Option D: Launching an extra-large notebook instance with 10+ TB of storage is costly and inefficient, especially when only a subset of the data is needed. Downloading the entire dataset to the notebook is also unnecessary.
Supporting Links:

AWS Secrets Manager: https://aws.amazon.com/secrets-manager/
Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Amazon Redshift: https://aws.amazon.com/redshift/
Amazon S3: https://aws.amazon.com/s3/
SageMaker Studio: https://aws.amazon.com/sagemaker/studio/

Question: 206

Exam Heist

A company is building an application that can predict spam email messages based on email text. The company can generate a few thousand human-labeled datasets that contain a list of email messages and a label of "spam" or "not spam" for each email message. A machine learning (ML) specialist wants to use transfer learning with a Bidirectional Encoder Representations from Transformers (BERT) model that is trained on English Wikipedia text data.

What should the ML specialist do to initialize the model to fine-tune the model with the custom data?

- Initialize the model with pretrained weights in all layers except the last fully connected layer.
- Initialize the model with pretrained weights in all layers. Stack a classifier on top of the first output position. Train the classifier with the labeled data.
- Initialize the model with random weights in all layers. Replace the last fully connected layer with a classifier. Train the classifier with the labeled data.
- Initialize the model with pretrained weights in all layers. Replace the last fully connected layer with a classifier. Train the classifier with the labeled data. ✓

Explanation:

The best approach to fine-tune a BERT model for spam detection, given a limited labeled dataset, is to leverage transfer learning effectively. Option D provides the most sensible strategy. Initializing the model with pretrained weights in all layers is crucial because BERT's pre-training on a massive corpus like English Wikipedia has already equipped it with a rich understanding of language nuances, grammar, and semantic relationships. These learned features are generally applicable and valuable even for a specific task like spam detection. Replacing the last fully connected layer is essential because the pre-trained BERT was likely trained for a different task (e.g., masked language modeling, next sentence prediction). The original output layer won't be suitable for classifying emails as "spam" or "not spam." Therefore, a new classifier, specifically designed for this binary classification task, needs to be added. This typically involves a fully connected layer followed by a sigmoid activation function to produce a probability score. Training only the new classifier layer with the labeled data, while freezing the other layers, would likely result in underfitting, as the model would not be able to adapt to the specific characteristics of the spam dataset. Conversely, training the entire model from scratch (option C) with random weights negates the benefits of transfer learning and would require significantly more data to achieve comparable performance. Retraining the entire model with the labeled spam dataset allows the pretrained weights to be fine-tuned to the specific characteristics of spam emails, while retaining the valuable general language understanding acquired during pre-training.
Therefore, option D is the optimal choice because it combines the power of pre-trained weights with task-specific fine-tuning, maximizing the utilization of the limited labeled data.
For further research:

Hugging Face Transformers documentation on fine-tuning: https://huggingface.co/transformers/training.html
BERT paper: https://arxiv.org/abs/1810.04805
Transfer learning concepts: https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-tensorflow-and-keras-b1c042450150

Question: 207

Exam Heist

A company is using a legacy telephony platform and has several years remaining on its contract. The company wants to move to AWS and wants to implement the following machine learning features:

•Call transcription in multiple languages
•Categorization of calls based on the transcript
•Detection of the main customer issues in the calls
•Customer sentiment analysis for each line of the transcript, with positive or negative indication and scoring of that sentiment

Which AWS solution will meet these requirements with the LEAST amount of custom model training?

- Use Amazon Transcribe to process audio calls to produce transcripts, categorize calls, and detect issues. Use Amazon Comprehend to analyze sentiment.
- Use Amazon Transcribe to process audio calls to produce transcripts. Use Amazon Comprehend to categorize calls, detect issues, and analyze sentiment
- Use Contact Lens for Amazon Connect to process audio calls to produce transcripts, categorize calls, detect issues, and analyze sentiment. ✓
- Use Contact Lens for Amazon Connect to process audio calls to produce transcripts. Use Amazon Comprehend to categorize calls, detect issues, and analyze sentiment.

Explanation:
The correct answer is C, using Contact Lens for Amazon Connect. Here's why:

Contact Lens for Amazon Connect is specifically designed to analyze contact center interactions, and its features closely align with the company's requirements. It natively supports call transcription, sentiment analysis, categorization, and issue detection, minimizing the need for custom model training.
Transcription: Contact Lens automatically transcribes calls.
Categorization & Issue Detection: Contact Lens can categorize calls based on keywords, topics, and sentiment trends, and can identify key customer issues using built-in natural language processing (NLP).
Sentiment Analysis: Contact Lens provides sentiment analysis at the utterance level, assigning positive or negative scores to each segment of the conversation.
Multi-language support: Contact Lens supports multiple languages for transcription and analysis.
Amazon Comprehend, while capable of sentiment analysis and categorization, lacks the direct integration with telephony systems that Contact Lens offers. Additionally, Comprehend requires additional configurations and coding to integrate with call transcripts.
Option A is incorrect because Amazon Transcribe primarily focuses on transcription and lacks built-in capabilities for call categorization and issue detection that are part of the Contact Lens service.
Option B requires the use of Amazon Transcribe to generate transcripts, then ingesting these into Amazon Comprehend for more advanced analysis, increasing integration complexity.
Option D is incorrect because Contact Lens alone can handle all of the listed requirements, while Comprehend only handles a portion.
Contact Lens integrates all necessary features within the Amazon Connect ecosystem, resulting in a streamlined, efficient, and low-code/no-code solution that fulfills all requirements with minimal custom training.
Supporting Links:

Contact Lens for Amazon Connect: https://aws.amazon.com/connect/contact-lens/

Question: 208

Exam Heist

A finance company needs to forecast the price of a commodity. The company has compiled a dataset of historical daily prices. A data scientist must train various forecasting models on 80% of the dataset and must validate the efficacy of those models on the remaining 20% of the dataset.

How should the data scientist split the dataset into a training dataset and a validation dataset to compare model performance?

- Pick a date so that 80% of the data points precede the date. Assign that group of data points as the training dataset. Assign all the remaining data points to the validation dataset. ✓
- Pick a date so that 80% of the data points occur after the date. Assign that group of data points as the training dataset. Assign all the remaining data points to the validation dataset.
- Starting from the earliest date in the dataset, pick eight data points for the training dataset and two data points for the validation dataset. Repeat this stratified sampling until no data points remain.
- Sample data points randomly without replacement so that 80% of the data points are in the training dataset. Assign all the remaining data points to the validation dataset.

Explanation:
The correct answer is A because time series data, like commodity prices, has a temporal dependence. This means the price at a given time is influenced by prices at previous times. Splitting the data randomly (as in option D) would introduce data leakage. The validation set would contain future data points influencing the training set, leading to an unrealistically optimistic evaluation of the model's performance. Options B and C don't properly consider the temporal ordering of the data. Option B trains on future data and validates on past data, which is illogical for forecasting. Option C performs a stratified sampling that, while potentially useful in other contexts, disregards the time series nature of the data and would likely lead to poor model performance due to inadequate representation of temporal patterns in the training and validation sets. Option A maintains the temporal order. The model is trained on past data and validated on future data, mimicking a real-world forecasting scenario. This provides a more realistic assessment of the model's ability to predict future prices based on past trends. Splitting based on date ensures no future data leaks into the training set, providing an accurate evaluation of the model's generalization capability on unseen, future data.
Here are some authoritative links to support this justification:

Time Series Forecasting: https://otexts.com/fpp2/
Data Leakage: https://machinelearningmastery.com/data-leakage-machine-learning/

Evaluating Time Series Models: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html (While this refers to sklearn's TimeSeriesSplit, it explains the importance of respecting time ordering)

Question: 209

Exam Heist

A retail company wants to build a recommendation system for the company's website. The system needs to provide recommendations for existing users and needs to base those recommendations on each user's past browsing history. The system also must filter out any items that the user previously purchased.

Which solution will meet these requirements with the LEAST development effort?

- Train a model by using a user-based collaborative filtering algorithm on Amazon SageMaker. Host the model on a SageMaker real-time endpoint. Configure an Amazon API Gateway API and an AWS Lambda function to handle real-time inference requests that the web application sends. Exclude the items that the user previously purchased from the results before sending the results back to the web application.
- Use an Amazon Personalize PERSONALIZED_RANKING recipe to train a model. Create a real-time filter to exclude items that the user previously purchased. Create and deploy a campaign on Amazon Personalize. Use the GetPersonalizedRanking API operation to get the real-time recommendations.
- Use an Amazon Personalize USER_PERSONALIZATION recipe to train a model. Create a real-time filter to exclude items that the user previously purchased. Create and deploy a campaign on Amazon Personalize. Use the GetRecommendations API operation to get the real-time recommendations. ✓
- Train a neural collaborative filtering model on Amazon SageMaker by using GPU instances. Host the model on a SageMaker real-time endpoint. Configure an Amazon API Gateway API and an AWS Lambda function to handle real-time inference requests that the web application sends. Exclude the items that the user previously purchased from the results before sending the results back to the web application.

Explanation:
The correct answer is C because it leverages Amazon Personalize's USER_PERSONALIZATION recipe, which is specifically designed for recommending items based on user interaction history, aligning directly with the requirement of basing recommendations on each user's past browsing history. Creating a real-time filter in Personalize efficiently handles the requirement of excluding previously purchased items. Furthermore, Personalize simplifies the deployment and inference process via campaigns and the `GetRecommendations` API. This reduces development effort compared to building and hosting a custom model on SageMaker (options A and D). Options A and D, involve significantly more manual work in model training, hosting, API creation, and filtering implementation, increasing development complexity. Option B uses the `PERSONALIZED_RANKING` recipe, which ranks a pre-selected list of items for a user, rather than generating a list of recommendations from the entire catalog based on browsing history as the question requires. USER_PERSONALIZATION is optimized to discover what a user might be interested in, making it superior for new item discovery compared to personalized ranking. Personalize's managed nature reduces the burden of infrastructure management and model optimization, further minimizing development effort. Therefore, option C delivers a complete recommendation solution tailored to the prompt's requirements with minimal code and operational overhead.
Relevant links for further research:

Amazon Personalize: https://aws.amazon.com/personalize/
Amazon Personalize Recipes: https://docs.aws.amazon.com/personalize/latest/dg/working-with-recipes.html
Amazon Personalize Filters: https://docs.aws.amazon.com/personalize/latest/dg/filters.html
Amazon Personalize GetRecommendations API:
https://docs.aws.amazon.com/personalize/latest/dg/API_Recommendations.html

Question: 210

Exam Heist

A bank wants to use a machine learning (ML) model to predict if users will default on credit card payments. The training data consists of 30,000 labeled records and is evenly balanced between two categories. For the model, an ML specialist selects the Amazon SageMaker built-in XGBoost algorithm and configures a SageMaker automatic hyperparameter optimization job with the Bayesian method. The ML specialist uses the validation accuracy as the objective metric.

When the bank implements the solution with this model, the prediction accuracy is 75%. The bank has given the ML specialist 1 day to improve the model in production.

Which approach is the FASTEST way to improve the model's accuracy?

- Run a SageMaker incremental training based on the best candidate from the current model's tuning job. Monitor the same metric that was used as the objective metric in the previous tuning, and look for improvements.
- Set the Area Under the ROC Curve (AUC) as the objective metric for a new SageMaker automatic hyperparameter tuning job. Use the same maximum training jobs parameter that was used in the previous tuning job.
- Run a SageMaker warm start hyperparameter tuning job based on the current model's tuning job. Use the same objective metric that was used in the previous tuning. ✓
- Set the F1 score as the objective metric for a new SageMaker automatic hyperparameter tuning job. Double the maximum training jobs parameter that was used in the previous tuning job.

Explanation:
The fastest way to improve the model's accuracy within a day is to leverage warm start hyperparameter tuning. Warm start in SageMaker leverages the knowledge gained from previous hyperparameter tuning jobs to accelerate the search for optimal hyperparameters in a new tuning job. Given the tight deadline of one day, this reuses the already explored parameter space from the initial tuning.
Option A, incremental training, focuses on retraining an existing model with new data. However, the primary issue here seems to be the hyperparameter tuning, not the data. Thus, it's less likely to yield significant improvements within the given timeframe.
Option B involves a new hyperparameter tuning job with AUC as the objective metric. While AUC is a valid metric, starting from scratch will consume time and resources, potentially missing the one-day deadline. Furthermore, simply changing the objective metric might not be the most efficient approach without leveraging previous tuning efforts.
Option D suggests using F1 score as the objective metric and doubling the maximum training jobs. While F1 score can be relevant for imbalanced datasets (although the question mentions evenly balanced categories) increasing training jobs significantly increases the time. The key here is the limited time. Doubling training jobs would likely exceed the one-day limit. Furthermore, discarding the previous tuning information is inefficient.
Option C, running a warm start hyperparameter tuning job, reuses information and accelerates the discovery of better hyperparameters compared to starting anew or focusing on incremental training. Keeping the same objective metric allows us to refine the search based on what we already know worked (or didn't work) in the previous tuning.
Therefore, the fastest approach is C.
Reference:

SageMaker Hyperparameter Tuning: Warm Start

Question: 211

Exam Heist

A data scientist has 20 TB of data in CSV format in an Amazon S3 bucket. The data scientist needs to convert the data to Apache Parquet format.

How can the data scientist convert the file format with the LEAST amount of effort?

- Use an AWS Glue crawler to convert the file format.
- Write a script to convert the file format. Run the script as an AWS Glue job. ✓
- Write a script to convert the file format. Run the script on an Amazon EMR cluster.
- Write a script to convert the file format. Run the script in an Amazon SageMaker notebook.

Explanation:
The optimal solution for converting a large CSV dataset (20 TB) to Parquet format in S3 with minimal effort involves leveraging AWS Glue.
Here's why option B is the best choice:

AWS Glue's Purpose: AWS Glue is a fully managed extract, transform, and load (ETL) service specifically designed for data preparation and transformation. Its purpose aligns perfectly with converting file formats.
Glue Jobs for ETL: Glue Jobs allow you to define and execute ETL tasks using scripts written in languages like Python or Scala. This enables the conversion process to be automated.
Direct S3 Integration: Glue natively integrates with S3. It can directly read CSV data from the bucket and write the converted Parquet data back to S3, eliminating the need to move or stage data manually.
Scalability and Management: Glue handles the underlying infrastructure and scaling of the ETL process. It automatically provisions the necessary resources to process the large dataset, relieving the data scientist from managing servers or clusters.
Cost-Effectiveness: Glue is generally cost-effective for such tasks. It avoids the overhead of managing an EMR cluster (option C) or a SageMaker notebook instance (option D), which would require more manual configuration and potentially higher costs.

Automation with Minimal Coding: By writing a script (using PySpark or Scala), the data scientist can perform the conversion process. Glue Jobs can be scheduled or triggered based on events.
Avoidance of Crawlers: While AWS Glue crawlers can be used for schema discovery, they are not primarily designed for converting file formats (option A). They are best suited for registering table definitions in the AWS Glue Data Catalog.
In contrast, option C (EMR cluster) requires manual cluster setup, configuration, and management. Option D (SageMaker notebook) is less suitable due to the limited resources available within a single notebook instance for processing 20 TB of data, and it may not be as cost-effective or scalable as Glue. Glue offers a dedicated ETL environment, simplifying the conversion workflow and minimizing the data scientist's operational burden.
Therefore, the answer B is correct because it provides the easiest, most scalable, and cost-effective way to convert 20 TB of data from CSV to Parquet format in S3.
Authoritative Links:

AWS Glue Documentation: https://aws.amazon.com/glue/
AWS Big Data Blog - Using AWS Glue for ETL: https://aws.amazon.com/blogs/big-data/tag/aws-glue/

Question: 212

Exam Heist

A company is building a pipeline that periodically retrains its machine learning (ML) models by using new streaming data from devices. The company's data engineering team wants to build a data ingestion system that has high throughput, durable storage, and scalability. The company can tolerate up to 5 minutes of latency for data ingestion. The company needs a solution that can apply basic data transformation during the ingestion process.

Which solution will meet these requirements with the MOST operational efficiency?

- Configure the devices to send streaming data to an Amazon Kinesis data stream. Configure an Amazon Kinesis Data Firehose delivery stream to automatically consume the Kinesis data stream, transform the data with an AWS Lambda function, and save the output into an Amazon S3 bucket. ✓
- Configure the devices to send streaming data to an Amazon S3 bucket. Configure an AWS Lambda function that is invoked by S3 event notifications to transform the data and load the data into an Amazon Kinesis data stream. Configure an Amazon Kinesis Data Firehose delivery stream to automatically consume the Kinesis data stream and load the output back into the S3 bucket.
- Configure the devices to send streaming data to an Amazon S3 bucket. Configure an AWS Glue job that is invoked by S3 event notifications to read the data, transform the data, and load the output into a new S3 bucket.
- Configure the devices to send streaming data to an Amazon Kinesis Data Firehose delivery stream. Configure an AWS Glue job that connects to the delivery stream to transform the data and load the output into an Amazon S3 bucket.

Explanation:
The most operationally efficient solution is option A, which uses Kinesis Data Streams and Kinesis Data Firehose with Lambda for transformation. This architecture provides a direct path for streaming data ingestion, transformation, and storage.
Here's a detailed justification:

High Throughput and Scalability: Kinesis Data Streams is designed for high-throughput, real-time data streaming. It can handle massive volumes of data from numerous devices. https://aws.amazon.com/kinesis/data-streams/
Durable Storage: Amazon S3 offers highly durable and scalable storage for the transformed data, crucial for ML model retraining. https://aws.amazon.com/s3/
Basic Data Transformation: Kinesis Data Firehose integrates seamlessly with AWS Lambda, enabling lightweight data transformations during the ingestion process without requiring a separate processing step.
https://aws.amazon.com/kinesis/data-firehose/
Latency Requirement: Kinesis Data Firehose allows for low-latency delivery of data to S3, meeting the 5-minute latency tolerance.
Operational Efficiency: This approach minimizes the number of services involved and automates the entire ingestion pipeline, simplifying management and reducing operational overhead. It leverages managed services, abstracting away infrastructure management tasks.
Why other options are less efficient:

Option B: Involves sending data directly to S3 initially, triggering Lambda, and then feeding that data to Kinesis. This adds unnecessary complexity. The Kinesis Firehose would load data back into S3, which is redundant.
Option C: While AWS Glue can perform transformations, relying solely on S3 event notifications and Glue jobs for streaming data ingestion is less efficient than using Kinesis. Glue is generally better suited for batch processing or more complex transformations.

Option D: Configuring devices to directly send data to Kinesis Data Firehose and then using a Glue job connecting to it for transformation is less efficient. Kinesis Data Firehose's built-in integration with AWS Lambda for transformation eliminates the need for a separate Glue job. Lambda provides transformation close to the source with minimal code.

Question: 213

Exam Heist

A retail company is ingesting purchasing records from its network of 20,000 stores to Amazon S3 by using Amazon Kinesis Data Firehose. The company uses a small, server-based application in each store to send the data to AWS over the internet. The company uses this data to train a machine learning model that is retrained each day. The company's data science team has identified existing attributes on these records that could be combined to create an improved model.

Which change will create the required transformed records with the LEAST operational overhead?

- Create an AWS Lambda function that can transform the incoming records. Enable data transformation on the ingestion Kinesis Data Firehose delivery stream. Use the Lambda function as the invocation target. ✓
- Deploy an Amazon EMR cluster that runs Apache Spark and includes the transformation logic. Use Amazon EventBridge (Amazon CloudWatch Events) to schedule an AWS Lambda function to launch the cluster each day and transform the records that accumulate in Amazon S3. Deliver the transformed records to Amazon S3.
- Deploy an Amazon S3 File Gateway in the stores. Update the in-store software to deliver data to the S3 File Gateway. Use a scheduled daily AWS Glue job to transform the data that the S3 File Gateway delivers to Amazon S3.
- Launch a fleet of Amazon EC2 instances that include the transformation logic. Configure the EC2 instances with a daily cron job to transform the records that accumulate in Amazon S3. Deliver the transformed records to Amazon S3.

Explanation:
The best approach is to leverage Kinesis Data Firehose's built-in data transformation capabilities using an AWS Lambda function (Option A). This solution offers the least operational overhead because it integrates seamlessly into the existing data ingestion pipeline.
Kinesis Data Firehose allows you to transform data in real-time before it's delivered to S3. By configuring data transformation within the Firehose stream and pointing it to a Lambda function, the transformation logic is automatically applied to each incoming record. This eliminates the need for separate batch processing, complex scheduling, or managing additional infrastructure.
The Lambda function can be written to perform the required attribute combinations for model improvement. The overhead is minimal because Lambda is a serverless compute service; you only pay for the compute time consumed. This approach is highly scalable and requires no manual provisioning or management of servers.
Alternatives like EMR (Option B), S3 File Gateway with Glue (Option C), and EC2 instances with cron jobs (Option D) introduce significant operational overhead, including cluster management, scheduling complexities, instance maintenance, and potential scalability issues. EMR, although powerful for large-scale data processing, is overkill for simple attribute transformations. S3 File Gateway is unnecessary because the data is already flowing into AWS. EC2 instances require continuous management and are less cost-effective than Lambda for intermittent processing.
Data transformation with Kinesis Data Firehose and Lambda offers a streamlined, cost-effective, and manageable solution for real-time data enrichment and preparation for machine learning.
Refer to these links for further information:

Kinesis Data Firehose Data Transformation: https://docs.aws.amazon.com/firehose/latest/dev/data-transformation.html
AWS Lambda: https://aws.amazon.com/lambda/

Question: 214

Exam Heist

A sports broadcasting company is planning to introduce subtitles in multiple languages for a live broadcast. The commentary is in English. The company needs the transcriptions to appear on screen in French or Spanish, depending on the broadcasting country. The transcriptions must be able to capture domain-specific terminology, names, and locations based on the commentary context. The company needs a solution that can support options to provide tuning data.

Which combination of AWS services and features will meet these requirements with the LEAST operational overhead? (Choose two.)

- [ ] Amazon Transcribe with custom vocabularies
- [x] Amazon Transcribe with custom language models ✓
- [ ] Amazon SageMaker Seq2Seq
- [ ] Amazon SageMaker with Hugging Face Speech2Text

☐ Amazon Translate ✓

Explanation:
The correct answer is **BE**.
Here's a detailed justification:

Amazon Transcribe is the optimal service for real-time speech-to-text transcription. It supports streaming audio and provides accurate transcriptions that are crucial for live subtitling. https://aws.amazon.com/transcribe/
Custom vocabularies in Amazon Transcribe allow you to specify domain-specific terms, names, and locations. By uploading a custom vocabulary file with these terms, you can significantly improve the accuracy of the transcriptions, reducing errors and enhancing the viewer experience. https://docs.aws.amazon.com/transcribe/latest/dg/custom-vocabulary.html
Custom language models in Amazon Transcribe takes this customization one step further. A language model is a statistical model that describes the probability of a given word sequence occurring in a language. A custom language model will provide increased performance because it can learn the particular language trends of your desired subject matter.
Amazon Translate efficiently translates the transcribed English text into French or Spanish in real-time.
https://aws.amazon.com/translate/
Other options and why they are not as optimal:

A: While custom vocabularies are helpful, they do not assist with language trends as much as custom language models.
C & D: Using Amazon SageMaker directly (with Seq2Seq models or Hugging Face Speech2Text) would involve significantly more operational overhead. You would need to manage model training, deployment, and scaling, which is more complex than using the managed Amazon Transcribe and Translate services. This negates the requirement for the *least* operational overhead.
Therefore, combining Amazon Transcribe with Custom Vocabularies, and Amazon Translate offers the most efficient and accurate solution with minimal operational overhead for generating real-time subtitles in multiple languages.

Question: 215

Exam Heist

A data scientist at a retail company is forecasting sales for a product over the next 3 months. After preliminary analysis, the data scientist identifies that sales are seasonal and that holidays affect sales. The data scientist also determines that sales of the product are correlated with sales of other products in the same category.

The data scientist needs to train a sales forecasting model that incorporates this information.

Which solution will meet this requirement with the LEAST development effort?

- ○ Use Amazon Forecast with Holidays featurization and the built-in autoregressive integrated moving average (ARIMA) algorithm to train the model.
- ○ Use Amazon Forecast with Holidays featurization and the built-in DeepAR+ algorithm to train the model. ✓
- ○ Use Amazon SageMaker Processing to enrich the data with holiday information. Train the model by using the SageMaker DeepAR built-in algorithm.
- ○ Use Amazon SageMaker Processing to enrich the data with holiday information. Train the model by using the Gluon Time Series (GluonTS) toolkit.

Explanation:
The correct answer is **B. Use Amazon Forecast with Holidays featurization and the built-in DeepAR+ algorithm to train the model.**
Here's a detailed justification:
Amazon Forecast is a fully managed service specifically designed for time-series forecasting. It significantly reduces the development effort required compared to building a forecasting solution from scratch using SageMaker. The problem description highlights seasonality, holiday effects, and correlations with other products as crucial factors. Amazon Forecast can natively handle these aspects.
Option B leverages the built-in Holidays featurization, which automatically incorporates holiday information into the model. This is a key requirement outlined in the problem statement. DeepAR+ is a supervised learning algorithm for time series forecasting. It's based on recurrent neural networks (RNNs) and is well-suited for handling complex time series data with seasonal patterns and dependencies. It also allows for related time series (sales of other products) to be used as covariates. The DeepAR+ algorithm is also capable of using category embeddings which can improve accuracy if the time series contain categories.
Option A, while using Amazon Forecast, suggests ARIMA. ARIMA models are less effective than DeepAR+ when dealing with complex seasonal patterns and covariates (related time series). ARIMA requires extensive manual tuning and feature engineering, increasing development effort.

Options C and D involve using Amazon SageMaker directly. While SageMaker offers flexibility, it requires significantly more effort to set up data processing pipelines, feature engineering, model training, and deployment compared to using the managed Amazon Forecast service. Using SageMaker Processing for holiday enrichment and then training a DeepAR model (either built-in or GluonTS) necessitates writing and managing custom code, increasing complexity and development time. GluonTS provides model building blocks and is designed for research, not deployment. Also, SageMaker does not natively understand holiday features.
Therefore, Option B offers the least development effort by utilizing Amazon Forecast's managed capabilities, built-in holiday handling, and the powerful DeepAR+ algorithm to address all the stated requirements: seasonality, holiday effects, and related time series correlation.
Authoritative Links:

Amazon Forecast: https://aws.amazon.com/forecast/
Amazon Forecast Holidays Featurization: https://docs.aws.amazon.com/forecast/latest/dg/holidays.html
Amazon Forecast DeepAR+ Algorithm: https://docs.aws.amazon.com/forecast/latest/dg/deepar.html
Amazon SageMaker: https://aws.amazon.com/sagemaker/

Question: 216

Exam Heist

A company is building a predictive maintenance model for its warehouse equipment. The model must predict the probability of failure of all machines in the warehouse. The company has collected 10,000 event samples within 3 months. The event samples include 100 failure cases that are evenly distributed across 50 different machine types.

How should the company prepare the data for the model to improve the model's accuracy?

- Adjust the class weight to account for each machine type.
- Oversample the failure cases by using the Synthetic Minority Oversampling Technique (SMOTE). ✓
- Undersample the non-failure events. Stratify the non-failure events by machine type.
- Undersample the non-failure events by using the Synthetic Minority Oversampling Technique (SMOTE).

Explanation:
The correct answer is B: Oversample the failure cases by using the Synthetic Minority Oversampling Technique (SMOTE).
Here's a detailed justification:
The problem describes a classic imbalanced dataset scenario: 10,000 samples with only 100 failures. This extreme imbalance can severely bias the model, causing it to be highly accurate in predicting the majority class (non-failures) but performing poorly in predicting the minority class (failures), which is the area of interest for predictive maintenance. The model might simply learn to always predict "no failure."
Option A, adjusting class weights, is a reasonable approach to address class imbalance. However, with such a significant imbalance (100 vs. 9,900), simply adjusting weights might not be sufficient to give the minority class enough influence during training. Furthermore, it does nothing to generate new representative samples. It only re-weights existing ones.
Option B, using SMOTE to oversample the failure cases, directly addresses the imbalance by creating synthetic data points for the minority class. SMOTE works by interpolating between existing minority class samples, generating new, similar samples. This helps the model learn the characteristics of the failure cases more effectively, improving its ability to identify potential failures. Oversampling failure events provides the model with a richer understanding of failure patterns, leading to better predictive accuracy. Specifically, it can help improve recall for the failure class.
Option C, undersampling the non-failure events while stratifying by machine type, could reduce the overall dataset size and potentially address the imbalance. However, undersampling can lead to information loss. By discarding a significant portion of the majority class, the model might miss important patterns and correlations present in the complete dataset. Stratifying by machine type might help preserve some machine-specific information, but information loss remains a significant drawback. Undersampling is generally less desirable than oversampling techniques like SMOTE when data is already limited.
Option D, undersampling the non-failure events using SMOTE, is conceptually flawed. SMOTE is designed for *oversampling* the minority class, not undersampling the majority class. Applying SMOTE to the majority class doesn't make sense and won't address the class imbalance problem.
Therefore, oversampling with SMOTE is the most suitable approach because it directly increases the representation of the minority class without losing information and generates synthetic, yet representative, data points. This leads to a more balanced and robust model.
Authoritative Links:

SMOTE Paper: Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16*, 321-357. https://www.jair.org/index.php/jair/article/view/1030/2408

Imbalanced Datasets: A general resource discussing the challenges of imbalanced datasets and various solutions: https://developers.google.com/machine-learning/data-prep/transform/imbalanced-data

Question: 217

Exam Heist

A company stores its documents in Amazon S3 with no predefined product categories. A data scientist needs to build a machine learning model to categorize the documents for all the company's products.

Which solution will meet these requirements with the MOST operational efficiency?

- Build a custom clustering model. Create a Dockerfile and build a Docker image. Register the Docker image in Amazon Elastic Container Registry (Amazon ECR). Use the custom image in Amazon SageMaker to generate a trained model.
- Tokenize the data and transform the data into tabular data. Train an Amazon SageMaker k-means model to generate the product categories.
- Train an Amazon SageMaker Neural Topic Model (NTM) model to generate the product categories. ✓
- Train an Amazon SageMaker Blazing Text model to generate the product categories.

Explanation:

The most operationally efficient solution is to use Amazon SageMaker Neural Topic Model (NTM). Here's why:

NTM is designed for topic modeling: NTM is specifically built to discover abstract "topics" within a collection of documents. These topics can naturally represent the product categories the company is looking for. This aligns directly with the problem statement.
SageMaker offers managed NTM: SageMaker provides a managed implementation of NTM, significantly reducing the operational overhead of setting up and managing the training infrastructure. This aligns with the "MOST operational efficiency" requirement.
Simplicity: NTM models accept raw text as input and directly produce topic assignments, minimizing preprocessing effort.
Other options are less suitable:

Option A (Custom Clustering): Building a custom clustering model using Docker and ECR involves much more manual effort in model development, containerization, and deployment. It's less operationally efficient than using a pre-built SageMaker algorithm.
Option B (K-means): K-means requires converting text to tabular data (e.g., using TF-IDF), which adds complexity. Also, K-means focuses on grouping similar data points based on distance, and doesn't natively extract topics from text like NTM.
Option D (BlazingText): BlazingText is primarily for word embedding and text classification, not topic modeling. It doesn't directly discover underlying topics in a collection of documents to categorize them into product types. While text classification could potentially be adapted, NTM offers a more direct and efficient approach for topic discovery.
Therefore, leveraging SageMaker's NTM model is the most streamlined and efficient way to categorize the company's documents into product categories.
Supporting links:

Amazon SageMaker NTM: https://docs.aws.amazon.com/sagemaker/latest/dg/ntm.html
Amazon SageMaker Built-in Algorithms: https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

Question: 218

Exam Heist

A sports analytics company is providing services at a marathon. Each runner in the marathon will have their race ID printed as text on the front of their shirt. The company needs to extract race IDs from images of the runners.

Which solution will meet these requirements with the LEAST operational overhead?

- Use Amazon Rekognition. ✓
- Use a custom convolutional neural network (CNN).
- Use the Amazon SageMaker Object Detection algorithm.
- Use Amazon Lookout for Vision.

Explanation:

The correct answer is A, using Amazon Rekognition. Here's why:

Amazon Rekognition offers a pre-trained optical character recognition (OCR) service specifically designed to extract text from images. This includes text appearing in various fonts, sizes, and orientations, which is crucial for recognizing race IDs printed on shirts. Critically, Rekognition eliminates the need to build, train, and manage a custom machine learning model.
Option B, using a custom CNN, would require significant operational overhead. This involves collecting and labeling a large dataset of race ID images, designing and training a complex neural network architecture, and continuously monitoring and retraining the model to maintain accuracy. This is a time-consuming and resource-intensive process.
Option C, using Amazon SageMaker Object Detection, is overkill. While object detection can identify the region containing the race ID, it still necessitates additional processing (like OCR) to actually extract the text. Rekognition combines both functionalities in a single, readily available service.
Option D, Amazon Lookout for Vision, is designed for anomaly detection in images, such as identifying defects in manufactured products. It is not suitable for OCR or extracting text information.
Rekognition's managed nature means AWS handles the underlying infrastructure, scaling, and maintenance. This minimizes operational responsibilities for the sports analytics company, making it the most efficient solution. The pay-as-you-go pricing model also ensures cost-effectiveness.
For further research, consult the official Amazon Rekognition documentation: https://aws.amazon.com/rekognition/ and specifically the text detection feature: https://docs.aws.amazon.com/rekognition/latest/dg/text-detection.html.

Question: 219

Exam Heist

A manufacturing company wants to monitor its devices for anomalous behavior. A data scientist has trained an Amazon SageMaker scikit-learn model that classifies a device as normal or anomalous based on its 4-day telemetry. The 4-day telemetry of each device is collected in a separate file and is placed in an Amazon S3 bucket once every hour. The total time to run the model across the telemetry for all devices is 5 minutes.

What is the MOST cost-effective solution for the company to use to run the model across the telemetry for all the devices?

- SageMaker Batch Transform ✓
- SageMaker Asynchronous Inference
- SageMaker Processing
- A SageMaker multi-container endpoint

Explanation:
Here's a detailed justification for why SageMaker Batch Transform is the most cost-effective solution for the manufacturing company's anomaly detection problem:
The core requirement is to analyze telemetry data stored in S3 for anomaly detection. The data arrives hourly, and the processing time for all devices is only 5 minutes. Cost-effectiveness is the primary concern.

SageMaker Batch Transform: This service is designed for offline inference on large datasets. It directly processes data in S3 and places the predictions back into S3. Since the inference time is relatively short (5 minutes) and the data is already in S3, Batch Transform provides an efficient and cost-effective solution. It avoids the overhead of maintaining a persistent endpoint. Batch Transform also scales automatically.
SageMaker Asynchronous Inference: This service is ideal for requests with large payload sizes, long processing times, or that require custom pre-and post-processing. While potentially suitable, the short processing time (5 minutes) and the fact that Batch Transform also handles data in S3 makes it less ideal than Batch Transform for cost-effectiveness. Asynchronous Inference is usually more appropriate for scenarios where latency is not critical but processing time is longer and the client doesn't want to wait for the inference to complete.
SageMaker Processing: This service is best suited for data pre-processing, feature engineering, model evaluation, and other data processing tasks. While it *could* be used to run the inference, it introduces unnecessary complexity. Processing jobs typically involve running custom scripts, which adds overhead compared to directly leveraging a pre-trained model via Batch Transform.
SageMaker Multi-Container Endpoint: This service hosts multiple models or different versions of the same model behind a single endpoint. It is used for real-time inference where low latency is crucial. Given the hourly batch processing requirement, maintaining a continuously running endpoint is less cost-effective than using Batch Transform. Multi-Container endpoints are specifically built to serve requests as soon as possible, a design that Batch Transform does not need.
Therefore, SageMaker Batch Transform provides the simplest, most direct, and cost-effective way to analyze the telemetry data stored in S3 on an hourly basis without the need for a continuously running endpoint or custom processing scripts. Batch Transform also avoids the cost of a persistent endpoint.
Authoritative Links:

Amazon SageMaker Batch Transform: https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html
Amazon SageMaker Asynchronous Inference: https://docs.aws.amazon.com/sagemaker/latest/dg/async-inference.html

Amazon SageMaker Processing: https://docs.aws.amazon.com/sagemaker/latest/dg/processing-job.html
Amazon SageMaker Multi-Container Endpoints: https://docs.aws.amazon.com/sagemaker/latest/dg/inference-multi-model-endpoints.html

Question: 220

Exam Heist

A company wants to segment a large group of customers into subgroups based on shared characteristics. The company's data scientist is planning to use the Amazon SageMaker built-in k-means clustering algorithm for this task. The data scientist needs to determine the optimal number of subgroups (k) to use.

Which data visualization approach will MOST accurately determine the optimal value of k?

- Calculate the principal component analysis (PCA) components. Run the k-means clustering algorithm for a range of k by using only the first two PCA components. For each value of k, create a scatter plot with a different color for each cluster. The optimal value of k is the value where the clusters start to look reasonably separated.
- Calculate the principal component analysis (PCA) components. Create a line plot of the number of components against the explained variance. The optimal value of k is the number of PCA components after which the curve starts decreasing in a linear fashion.
- Create a t-distributed stochastic neighbor embedding (t-SNE) plot for a range of perplexity values. The optimal value of k is the value of perplexity, where the clusters start to look reasonably separated.
- Run the k-means clustering algorithm for a range of k. For each value of k, calculate the sum of squared errors (SSE). Plot a line chart of the SSE for each value of k. The optimal value of k is the point after which the curve starts decreasing in a linear fashion. ✓

Explanation:
The correct answer is D because it directly addresses the problem of determining the optimal 'k' for k-means clustering using the Elbow Method, a standard practice for this purpose. The Elbow Method involves plotting the Sum of Squared Errors (SSE) against different values of 'k'. The optimal 'k' is identified as the point where the decrease in SSE starts to diminish significantly, visually resembling an "elbow" in the plot. This point suggests that adding more clusters provides diminishing returns in terms of reducing within-cluster variance.
Option A is incorrect because while PCA can reduce dimensionality and help visualize data, it doesn't directly determine the optimal 'k' for clustering. Visual inspection of scatter plots with the first two PCA components is subjective and can be misleading, especially with high-dimensional data.
Option B is incorrect because while PCA can help determine the number of principal components to use, it doesn't identify the optimal number of clusters ('k'). The explained variance plot helps to identify the components that capture the most variance in the data but doesn't consider the specific clustering objective.
Option C is incorrect because t-SNE is a dimensionality reduction technique focused on visualizing high-dimensional data in a low-dimensional space (usually 2D or 3D), but is mainly for visualization and not directly suited for determining the optimal 'k' for k-means. While t-SNE can reveal cluster structure, its interpretation is complex and its parameter, perplexity, doesn't directly translate to the number of clusters.
Therefore, option D is the most appropriate because it uses the Elbow Method, a well-established technique for determining the optimal number of clusters in k-means, by directly observing the decrease in SSE as 'k' increases.
Further Research:

Elbow Method: https://en.wikipedia.org/wiki/Elbow_method_(clustering)
K-means Clustering: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Question: 221

Exam Heist

A data scientist at a financial services company used Amazon SageMaker to train and deploy a model that predicts loan defaults. The model analyzes new loan applications and predicts the risk of loan default. To train the model, the data scientist manually extracted loan data from a database. The data scientist performed the model training and deployment steps in a Jupyter notebook that is hosted on SageMaker Studio notebooks. The model's prediction accuracy is decreasing over time.

Which combination of steps is the MOST operationally efficient way for the data scientist to maintain the model's accuracy? (Choose two.)

- [x] Use SageMaker Pipelines to create an automated workflow that extracts fresh data, trains the model, and deploys a new version of the model. ✓
- [x] Configure SageMaker Model Monitor with an accuracy threshold to check for model drift. Initiate an Amazon CloudWatch alarm when the threshold is exceeded. Connect the workflow in SageMaker Pipelines with the CloudWatch alarm to automatically initiate retraining. ✓
- [ ] Store the model predictions in Amazon S3. Create a daily SageMaker Processing job that reads the predictions from Amazon S3, checks for changes in model prediction accuracy, and sends an email notification if a significant change is detected.

- [ ] Rerun the steps in the Jupyter notebook that is hosted on SageMaker Studio notebooks to retrain the model and redeploy a new version of the model.
- [ ] Export the training and deployment code from the SageMaker Studio notebooks into a Python script. Package the script into an Amazon Elastic Container Service (Amazon ECS) task that an AWS Lambda function can initiate.

Explanation:
The correct answer is AB. Here's why:

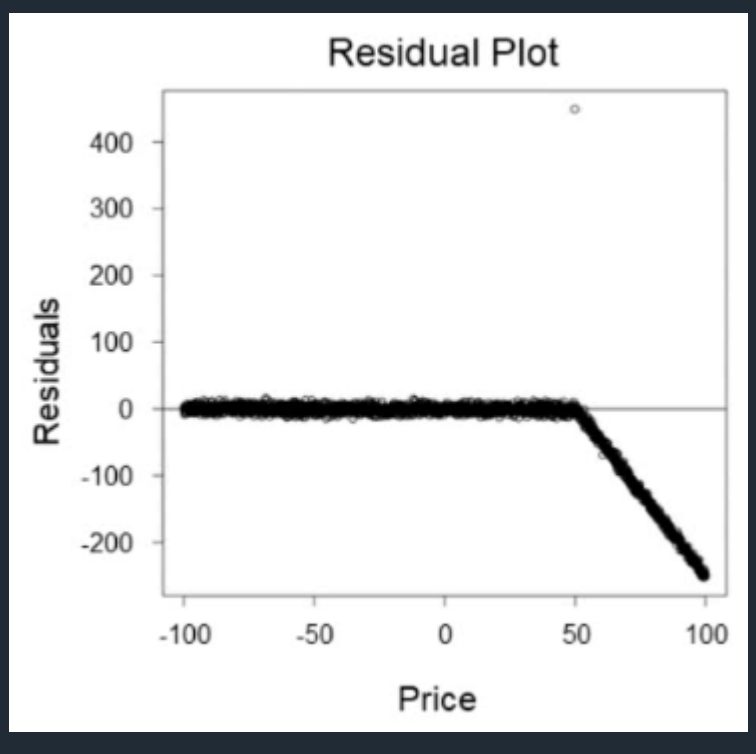
A. Use SageMaker Pipelines to create an automated workflow...: SageMaker Pipelines is designed for creating robust, repeatable, and auditable machine learning workflows. By automating the data extraction, training, and deployment processes, the data scientist eliminates manual intervention, reduces the risk of errors, and ensures that the model is regularly retrained with the latest data. This is crucial for maintaining accuracy as the underlying data distribution might shift over time (a phenomenon known as data drift). An automated pipeline addresses the operational efficiency concern.
https://aws.amazon.com/sagemaker/pipelines/
B. Configure SageMaker Model Monitor with an accuracy threshold...: Model Monitor is a feature of SageMaker specifically designed to detect model drift in production. By setting an accuracy threshold, the system can automatically detect when the model's performance degrades. Linking this to a CloudWatch alarm enables automated notifications and, more importantly, can trigger the retraining process defined in the SageMaker Pipeline. This ensures that the model is automatically updated when needed, addressing the problem of decreasing accuracy and the need for proactive maintenance.
https://aws.amazon.com/sagemaker/model-monitor/
Why other options are less ideal:

C. Store the model predictions in Amazon S3...: While storing predictions can be useful for analysis, creating a daily SageMaker Processing job that only sends an email notification is a reactive approach. It doesn't automatically retrain the model, requiring manual intervention. It is less operationally efficient.
D. Rerun the steps in the Jupyter notebook...: Rerunning the notebook manually is not scalable or operationally efficient in the long term. It's prone to errors and doesn't address the need for continuous monitoring and retraining. It is essentially repeating the manual process that initially caused the problem.
E. Export the training and deployment code...: While this is an improvement over the initial notebook-based approach, using ECS tasks and Lambda functions for retraining is more complex to set up and manage compared to SageMaker Pipelines. SageMaker Pipelines are designed specifically for end-to-end ML workflows and provide better integration with other SageMaker features like Model Monitor.
In summary, SageMaker Pipelines and Model Monitor offer a cohesive and automated solution for maintaining model accuracy by continuously monitoring performance and automatically retraining the model when necessary, making it the most operationally efficient approach.

Question: 222

Exam Heist

A retail company wants to create a system that can predict sales based on the price of an item. A machine learning (ML) engineer built an initial linear model that resulted in the following residual plot:



Which actions should the ML engineer take to improve the accuracy of the predictions in the next phase of model building? (Choose three.)

- [ ] Downsample the data uniformly to reduce the amount of data.

- [x] Create two different models for different sections of the data. ✓
- [ ] Downsample the data in sections where Price < 50.
- [ ] Offset the input data by a constant value where Price > 50.
- [x] Examine the input data, and apply non-linear data transformations where appropriate. ✓
- [x] Use a non-linear model instead of a linear model. ✓

Explanation:

-B. Create two different models for different sections of the data-If the dataset exhibits distinct patterns in different sections, training separate models can help capture unique trends more effectively.

-E. Examine the input data, and apply non-linear data transformations where appropriate-Non-linear transformations can help reveal hidden relationships in the data, improving model performance.

-F. Use a non-linear model instead of a linear model-If the data exhibits complex relationships, a non-linear model can better capture variations and improve accuracy.

Question: 223 Exam Heist

A data scientist at a food production company wants to use an Amazon SageMaker built-in model to classify different vegetables. The current dataset has many features. The company wants to save on memory costs when the data scientist trains and deploys the model. The company also wants to be able to find similar data points for each test data point.

Which algorithm will meet these requirements?

- ◉ K-nearest neighbors (k-NN) with dimension reduction ✓
- ○ Linear learner with early stopping
- ○ K-means
- ○ Principal component analysis (PCA) with the algorithm mode set to random

Explanation:

Here's a detailed justification for why K-Nearest Neighbors (k-NN) with dimension reduction is the most suitable algorithm for the described scenario:

The problem statement emphasizes two primary goals: memory cost reduction during training and deployment, and the ability to find similar data points (neighbors) for test data.

k-NN for Similarity Search: k-NN is inherently designed for finding the *k* closest data points to a given query. This directly addresses the requirement of finding similar data points. It works by calculating the distance between the test point and all training points and selecting the *k* nearest ones.

Dimension Reduction for Memory Efficiency: The dataset has many features, leading to high memory consumption. Dimension reduction techniques, such as Principal Component Analysis (PCA) or feature selection, can significantly reduce the number of features while preserving most of the data's variance. This directly addresses the need to save on memory costs. Combining k-NN with dimension reduction optimizes memory usage during both training and deployment, as the model will work with fewer features.

Why other options are less suitable:

Linear Learner with Early Stopping: While early stopping can prevent overfitting, linear learners might not be well-suited for complex, non-linear relationships in the data. Additionally, linear learners don't natively provide a mechanism for finding similar data points in the way k-NN does.

K-Means: k-means is a clustering algorithm, used for grouping similar data points into clusters. It does not classify vegetables or find nearest neighbors in the classification sense, and it is not designed for classification tasks directly.

Principal Component Analysis (PCA) with the algorithm mode set to random: PCA can be used for dimension reduction. However, PCA itself is not a classification algorithm. It must be followed by another algorithm to perform classification. Combining PCA alone won't achieve the goal of classifying vegetables *and* identifying similar data points. The "random" algorithm mode in PCA isn't standard.

Therefore, the best solution is to use k-NN for its inherent nearest neighbor search capability and combine it with dimension reduction techniques to reduce memory costs.

Supporting Documentation:

Amazon SageMaker k-NN: https://docs.aws.amazon.com/sagemaker/latest/dg/knn_hyperparameters.html
Dimension Reduction: https://scikit-learn.org/stable/modules/unsupervised_reduction.html
Principal Component Analysis (PCA): https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Question: 224

Exam Heist

A data scientist is training a large PyTorch model by using Amazon SageMaker. It takes 10 hours on average to train the model on GPU instances. The data scientist suspects that training is not converging and that resource utilization is not optimal.

What should the data scientist do to identify and address training issues with the LEAST development effort?

- Use CPU utilization metrics that are captured in Amazon CloudWatch. Configure a CloudWatch alarm to stop the training job early if low CPU utilization occurs.
- Use high-resolution custom metrics that are captured in Amazon CloudWatch. Configure an AWS Lambda function to analyze the metrics and to stop the training job early if issues are detected.
- Use the SageMaker Debugger vanishing_gradient and LowGPUUtilization built-in rules to detect issues and to launch the StopTrainingJob action if issues are detected. ✓
- Use the SageMaker Debugger confusion and feature_importance_overweight built-in rules to detect issues and to launch the StopTrainingJob action if issues are detected.

Explanation:
The best approach to identify and address the training issues with the least development effort is option C, using SageMaker Debugger with built-in rules like `vanishing_gradient` and `LowGPUUtilization` and leveraging its `StopTrainingJob` action. SageMaker Debugger is designed specifically for identifying training issues in machine learning models. It provides built-in rules that automatically detect common problems, such as vanishing gradients (which could explain the lack of convergence) and low GPU utilization (which indicates suboptimal resource usage). The `vanishing_gradient` rule directly addresses the suspicion that the training isn't converging. The `LowGPUUtilization` rule addresses the suspicion of suboptimal resource utilization.
Furthermore, SageMaker Debugger integrates directly with SageMaker training jobs. Configuring the built-in rules to trigger the `StopTrainingJob` action enables automatic termination of inefficient or problematic training runs, minimizing wasted resources and time. This "out-of-the-box" functionality minimizes the development effort required compared to custom solutions.
Option A, relying solely on CPU utilization metrics, is insufficient because CPU utilization is not directly indicative of model convergence or efficient GPU usage. Option B, using custom high-resolution metrics and a Lambda function, requires significant development and configuration effort to define the metrics, write the Lambda function for analysis, and set up triggers. This is far more complex than using SageMaker Debugger's built-in capabilities. Option D utilizes rules (`confusion` and `feature_importance_overweight`) that are more relevant for model evaluation and feature analysis *after* training, not for detecting convergence or resource utilization issues *during* training. Therefore, option C offers the most efficient and relevant solution.
Amazon SageMaker Debugger documentationSageMaker Debugger built-in rules

Question: 225

Exam Heist

A bank wants to launch a low-rate credit promotion campaign. The bank must identify which customers to target with the promotion and wants to make sure that each customer's full credit history is considered when an approval or denial decision is made.

The bank's data science team used the XGBoost algorithm to train a classification model based on account transaction features. The data science team deployed the model by using the Amazon SageMaker model hosting service. The accuracy of the model is sufficient, but the data science team wants to be able to explain why the model denies the promotion to some customers.

What should the data science team do to meet this requirement in the MOST operationally efficient manner?

- Create a SageMaker notebook instance. Upload the model artifact to the notebook. Use the plot_importance() method in the Python XGBoost interface to create a feature importance chart for the individual predictions.
- Retrain the model by using SageMaker Debugger. Configure Debugger to calculate and collect Shapley values. Create a chart that shows features and SHapley. Additive explanations (SHAP) values to explain how the features affect the model outcomes.
- Set up and run an explainability job powered by SageMaker Clarify to analyze the individual customer data, using the training data as a baseline. Create a chart that shows features and SHapley Additive explanations (SHAP) values to explain how the features affect the model outcomes. ✓

- Use SageMaker Model Monitor to create Shapley values that help explain model behavior. Store the Shapley values in Amazon S3. Create a chart that shows features and SHapley Additive explanations (SHAP) values to explain how the features affect the model outcomes.

Explanation:

The correct answer is **C**. Here's why:

The requirement is to explain individual prediction outcomes of an XGBoost model deployed on SageMaker, in an operationally efficient manner. This calls for a dedicated explainability solution tightly integrated with SageMaker.

Option A is incorrect because using `plot_importance()` in a SageMaker notebook creates *global* feature importance, showing the overall contribution of each feature during training. It doesn't provide explanations for individual customer prediction outcomes, and it involves manual work within a notebook, making it less operationally efficient for production use.

Option B involves retraining the model with SageMaker Debugger to calculate Shapley values. While Debugger can compute Shapley values, it's primarily for debugging and profiling during training, not for generating explanations for individual predictions in a deployed model serving real-time requests. Retraining is also a costly and time-consuming process.

Option C, leveraging SageMaker Clarify, directly addresses the requirement. SageMaker Clarify is specifically designed for model explainability. It can analyze individual customer data and calculate Shapley values (SHAP) to provide feature importance for each prediction. Using the training data as a baseline helps understand how each feature pushed the prediction one way or the other relative to the overall population. This is a built-in feature and therefore the most efficient operationally.

Option D suggests using SageMaker Model Monitor. Model Monitor is designed to detect data drift and model performance degradation over time. While it's valuable for production model monitoring, it doesn't natively calculate Shapley values or provide individual prediction explanations. It focuses on tracking overall model health, not explaining specific predictions.

Therefore, SageMaker Clarify offers a streamlined, integrated, and scalable approach to generate Shapley values and explain individual customer outcomes in a production environment, making it the most operationally efficient option.

Authoritative links:

SageMaker Clarify: https://aws.amazon.com/sagemaker/clarify/
SageMaker Clarify Documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-processing-job.html

Question: 226

Exam Heist

A company has hired a data scientist to create a loan risk model. The dataset contains loan amounts and variables such as loan type, region, and other demographic variables. The data scientist wants to use Amazon SageMaker to test bias regarding the loan amount distribution with respect to some of these categorical variables.

Which pretraining bias metrics should the data scientist use to check the bias distribution? (Choose three.)

- [ ] Class imbalance
- [ ] Conditional demographic disparity
- [ ] Difference in proportions of labels
- [x] Jensen-Shannon divergence ✓
- [x] Kullback-Leibler divergence ✓
- [x] Total variation distance ✓

Explanation:

Here's a detailed justification for why options D, E, and F (Jensen-Shannon divergence, Kullback-Leibler divergence, and Total variation distance) are the correct pretraining bias metrics for assessing loan amount distribution disparities with respect to categorical variables, while A, B, and C are not.

The core task is to determine if loan amounts are distributed differently across various demographic groups (defined by the categorical variables). This involves comparing probability distributions. Options D, E, and F are metrics specifically designed to quantify the similarity or difference between probability distributions.

D. Jensen-Shannon Divergence (JSD): JSD measures the similarity between two probability distributions. It is a symmetrized and smoothed version of the Kullback-Leibler divergence. In this context, the data scientist would calculate the loan amount distribution for each demographic group (e.g., loan amount distribution for each region) and then use JSD to compare those distributions pairwise. A high JSD indicates a significant difference in loan amount distributions across the regions, revealing potential bias. https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon_divergence

E. Kullback-Leibler Divergence (KL Divergence): KL divergence, also known as relative entropy, measures how one probability distribution diverges from a second, expected probability distribution. It's not symmetric. In this case, KL divergence would measure how much the loan amount distribution for one group differs from a reference group's distribution. A large KL

divergence indicates a significant difference and potential bias.
https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
F. Total Variation Distance (TVD): TVD measures the largest possible difference between the probabilities that two probability distributions can assign to the same event. In the context of loan amounts, TVD directly quantifies the maximum difference in probability between the loan amount distributions of different demographic groups. It offers a straightforward interpretation of the magnitude of the difference. https://en.wikipedia.org/wiki/Total_variation_distance_of_probability_measures
Now, consider why the other options are incorrect:

A. Class Imbalance: Class imbalance refers to an unequal distribution of classes in a classification problem (e.g., more approved loans than denied loans). While class imbalance can affect model performance, it doesn't directly address the distribution of loan *amounts* across different demographic groups, which is the focus here.
B. Conditional Demographic Disparity: This typically applies when examining disparities in model *outcomes* (e.g., loan approval rates) conditioned on a demographic attribute. While relevant for post-training bias evaluation, it's not the correct metric for examining the pre-training distribution of loan amounts with respect to demographic features.
C. Difference in Proportions of Labels: Similar to option B, this focuses on comparing the proportions of different *outcomes* or labels (e.g., loan approval rates) across different demographic groups. It's not suitable for analyzing the distribution of a continuous variable like loan amount.
Therefore, D, E, and F are specifically designed to compare probability distributions, making them suitable for detecting bias in the distribution of loan amounts across different categorical groups within the dataset before model training. They directly quantify the difference in distributions. SageMaker clarifies that distribution comparison is critical for uncovering such data bias before model creation.

Question: 227

Exam Heist

A retail company wants to use Amazon Forecast to predict daily stock levels of inventory. The cost of running out of items in stock is much higher for the company than the cost of having excess inventory. The company has millions of data samples for multiple years for thousands of items. The company's purchasing department needs to predict demand for 30-day cycles for each item to ensure that restocking occurs.

A machine learning (ML) specialist wants to use item-related features such as "category," "brand," and "safety stock count." The ML specialist also wants to use a binary time series feature that has "promotion applied?" as its name. Future promotion information is available only for the next 5 days.

The ML specialist must choose an algorithm and an evaluation metric for a solution to produce prediction results that will maximize company profit.

Which solution will meet these requirements?

- Train a model by using the Autoregressive Integrated Moving Average (ARIMA) algorithm. Evaluate the model by using the Weighted Quantile Loss (wQL) metric at 0.75 (P75).
- Train a model by using the Autoregressive Integrated Moving Average (ARIMA) algorithm. Evaluate the model by using the Weighted Absolute Percentage Error (WAPE) metric.
- Train a model by using the Convolutional Neural Network - Quantile Regression (CNN-QR) algorithm. Evaluate the model by using the Weighted Quantile Loss (wQL) metric at 0.75 (P75). ✓
- Train a model by using the Convolutional Neural Network - Quantile Regression (CNN-QR) algorithm. Evaluate the model by using the Weighted Absolute Percentage Error (WAPE) metric.

Explanation:
Here's a detailed justification for why option C is the correct answer:
The problem requires accurate demand forecasting to minimize stockouts, which are more costly than excess inventory. This necessitates an algorithm capable of handling complex data, including item-related features and a short-term binary time series feature ("promotion applied?"). The forecasting horizon is 30 days, but future promotion data is available for only 5 days. This aspect influences algorithm selection.
Why CNN-QR is appropriate:

Handles related time series and item metadata: CNN-QR excels in leveraging related time series data (millions of samples for thousands of items) and item-related features (category, brand, safety stock count) to improve forecast accuracy. It uses convolutional layers to automatically extract relevant patterns from the data.
https://docs.aws.amazon.com/forecast/latest/dg/aws-forecast-algorithms.html describes CNN-QR's ability to incorporate related time series and item metadata.
Quantile Regression for Cost Optimization: The CNN-QR algorithm uses quantile regression. Quantile regression directly estimates the conditional quantiles of the target variable (demand) given the input features. By optimizing for a higher quantile (like P75, which predicts a value that is exceeded only 25% of the time), we can bias the forecast towards higher demand, thereby reducing the risk of stockouts.

Handles short-term feature importance: While promotion data is only available for 5 days, CNN-QR can still learn the impact of this feature within that limited window and improve the accuracy of forecasts during those promotion periods.
Why wQL at 0.75 is appropriate:

Asymmetric Cost Function: wQL (Weighted Quantile Loss) is ideal when the cost of under-prediction (stockouts) is different from the cost of over-prediction (excess inventory).
Bias Towards Higher Demand: Setting wQL at 0.75 (P75) means the model will be penalized more heavily for under-predicting demand than for over-predicting it. This aligns with the business requirement of prioritizing the avoidance of stockouts over minimizing excess inventory. A higher quantile level ensures a more conservative (higher) forecast.
Optimizing for Profit: By using wQL at 0.75, the model aims to minimize the overall cost associated with both stockouts and excess inventory, ultimately maximizing company profit.
Why other options are incorrect:

ARIMA Limitations: ARIMA is a time series forecasting method that primarily relies on historical data patterns of a single time series. It struggles to effectively incorporate item-related features and related time series data from thousands of items. Therefore, options A and B are not appropriate.
WAPE Limitations: WAPE (Weighted Absolute Percentage Error) measures the overall accuracy of the forecast, but it doesn't consider the asymmetric costs associated with stockouts versus excess inventory. Using WAPE wouldn't directly help in minimizing the impact of costly stockouts.
In conclusion, CNN-QR with wQL at 0.75 is the most suitable solution because it effectively leverages related time series and item-related features, and it optimizes the forecast to minimize the cost of stockouts, which is the primary business goal.

Question: 228

Exam Heist

An online retail company wants to develop a natural language processing (NLP) model to improve customer service. A machine learning (ML) specialist is setting up distributed training of a Bidirectional Encoder Representations from Transformers (BERT) model on Amazon SageMaker. SageMaker will use eight compute instances for the distributed training.

The ML specialist wants to ensure the security of the data during the distributed training. The data is stored in an Amazon S3 bucket.

Which combination of steps should the ML specialist take to protect the data during the distributed training? (Choose three.)

- [x] Run distributed training jobs in a private VPC. Enable inter-container traffic encryption. ✓
- [ ] Run distributed training jobs across multiple VPCs. Enable VPC peering.
- [x] Create an S3 VPC endpoint. Then configure network routes, endpoint policies, and S3 bucket policies. ✓
- [x] Grant read-only access to SageMaker resources by using an IAM role. ✓
- [ ] Create a NAT gateway. Assign an Elastic IP address for the NAT gateway.
- [ ] Configure an inbound rule to allow traffic from a security group that is associated with the training instances.

Explanation:
Here's a breakdown of why the chosen answer (ACD) is correct and why the other options are less suitable for securing data during distributed training of a BERT model on SageMaker:

A. Run distributed training jobs in a private VPC. Enable inter-container traffic encryption: Running training jobs within a private VPC (Virtual Private Cloud) isolates the training environment from the public internet, significantly reducing the attack surface. Inter-container traffic encryption ensures that data transmitted between the containers running on different instances during the distributed training process is protected from eavesdropping and tampering. This is especially crucial for sensitive data like customer data.
C. Create an S3 VPC endpoint. Then configure network routes, endpoint policies, and S3 bucket policies: An S3 VPC endpoint allows SageMaker training instances within the VPC to access the S3 bucket containing the training data without traversing the public internet. Endpoint policies and S3 bucket policies provide granular control over who or what can access the data stored in S3. Using endpoint policies, you can specify which principals (e.g., the SageMaker IAM role) can access the S3 bucket.
D. Grant read-only access to SageMaker resources by using an IAM role: IAM (Identity and Access Management) roles are used to grant permissions to AWS services. Giving the SageMaker resources a read-only IAM role limits the potential damage if a compromise occurs. The training jobs only need to read the data from S3, write permissions should be avoided. This principle of least privilege is a cornerstone of cloud security.
Let's examine why the other options are not as appropriate:

B. Run distributed training jobs across multiple VPCs. Enable VPC peering: While VPC peering can be used for networking, it is overly complex and unnecessary for this scenario. It is better to contain the training job into a single VPC for easier security management.
E. Create a NAT gateway. Assign an Elastic IP address for the NAT gateway: A NAT (Network Address Translation) gateway allows instances in a private subnet to connect to the internet, but doesn't allow the internet to initiate connections with those instances. The training job does not require access to the internet and adding a NAT Gateway would only increase the attack surface of the VPC.
F. Configure an inbound rule to allow traffic from a security group that is associated with the training instances: While security groups are important for controlling network traffic, this step alone does not address the security of data stored in S3 or data in transit between containers. It's only a part of the security solution.
Authoritative Links:

Amazon VPC: https://aws.amazon.com/vpc/
VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
IAM Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
S3 Bucket Policies: https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html
SageMaker Security: https://docs.aws.amazon.com/sagemaker/latest/dg/security.html

Question: 229

Exam Heist

An analytics company has an Amazon SageMaker hosted endpoint for an image classification model. The model is a custom-built convolutional neural network (CNN) and uses the PyTorch deep learning framework. The company wants to increase throughput and decrease latency for customers that use the model.

Which solution will meet these requirements MOST cost-effectively?

- Use Amazon Elastic Inference on the SageMaker hosted endpoint. ✓
- Retrain the CNN with more layers and a larger dataset.
- Retrain the CNN with more layers and a smaller dataset.
- Choose a SageMaker instance type that has multiple GPUs.

Explanation:
The most cost-effective solution for increasing throughput and decreasing latency of an image classification model hosted on a SageMaker endpoint is to use Amazon Elastic Inference (EI). Here's why:
Option A leverages EI, which allows you to attach low-cost GPU-powered acceleration to existing Amazon EC2 instances, SageMaker instances, or AWS Lambda functions. For a PyTorch-based CNN, this is particularly advantageous. EI optimizes deep learning inference performance without requiring you to change your instance type or retrain your model. It is designed for situations where the GPU requirements for inference are lower than those for training, which is common in production deployments. This makes it significantly more cost-effective than provisioning a whole GPU instance when you only need a fraction of its computational power for inference.
Options B and C involve retraining the CNN with modified layers and datasets. Retraining is time-consuming and computationally expensive. Moreover, changing the model architecture might not necessarily improve inference latency; it might even worsen it if the model becomes more complex. The problem is not necessarily with the model, but with the inference serving capacity.
Option D, choosing a SageMaker instance type with multiple GPUs, is an option, but less cost-effective than EI. While multiple GPUs can increase throughput, they also significantly increase the instance cost. The model might not even fully utilize all the available GPUs, leading to wasted resources. EI provides a more granular and cost-optimized way to add GPU acceleration specifically for inference workloads. EI provides a lower-cost alternative to using full GPU instances for inference.
Therefore, using Amazon Elastic Inference allows the analytics company to scale up their inference capacity by adding the required GPU resources for inference without the need for a full GPU instance or expensive model retraining. This directly addresses both throughput and latency concerns in a cost-optimized manner.
Further Research:

Amazon Elastic Inference: https://aws.amazon.com/elastic-inference/
SageMaker Inference: https://aws.amazon.com/sagemaker/inference/

Question: 230

Exam Heist

An ecommerce company is collecting structured data and unstructured data from its website, mobile apps, and IoT devices. The data is stored in several databases and Amazon S3 buckets. The company is implementing a scalable repository to store structured data and unstructured data. The company must implement a solution that provides a central data catalog, self-service access to the data, and granular data access policies and encryption to protect the data.

Which combination of actions will meet these requirements with the LEAST amount of setup? (Choose three.)

- [x] Identify the existing data in the databases and S3 buckets. Link the data to AWS Lake Formation. ✓
- [ ] Identify the existing data in the databases and S3 buckets. Link the data to AWS Glue.
- [x] Run AWS Glue crawlers on the linked data sources to create a central data catalog. ✓
- [ ] Apply granular access policies by using AWS Identity and Access Management (1AM). Configure server-side encryption on each data source.
- [x] Apply granular access policies and encryption by using AWS Lake Formation. ✓
- [ ] Apply granular access policies and encryption by using AWS Glue.

Explanation:
The combination of actions **A, C, and E** provides the least amount of setup to meet the requirements of a central data catalog, self-service access, and granular data access/encryption.
Here's the justification:

A: Identify the existing data and link to AWS Lake Formation: Lake Formation needs to know where the data resides. Identifying existing databases and S3 buckets is the initial step. Linking data to Lake Formation makes the data sources discoverable.
C: Run AWS Glue crawlers: AWS Glue crawlers automatically discover the schema of the data in the linked sources. This schema information is populated in the AWS Glue Data Catalog, acting as the central metadata repository. This eliminates the need for manual schema definition and metadata management.
E: Apply granular access policies and encryption using AWS Lake Formation: Lake Formation provides fine-grained access control at the table and column level, allowing you to define who can access what data. Additionally, Lake Formation can enforce encryption on data at rest and in transit, enhancing data security. This centralizes access control and encryption management within Lake Formation.
Let's examine why other options are less ideal:

B: AWS Glue alone doesn't provide the granular access control features offered by Lake Formation. While Glue crawlers are useful, they only address the data cataloging aspect.
D: IAM policies applied directly to S3 buckets and databases are less manageable and scalable compared to Lake Formation's centralized approach. Configuring server-side encryption on each data source separately is more work than managing it through Lake Formation.
F: AWS Glue focuses on ETL (Extract, Transform, Load) and data cataloging but lacks the comprehensive data governance and security features present in Lake Formation.
In summary, Lake Formation builds upon AWS Glue Data Catalog to offer a centralized data governance solution, including granular access control and encryption. By linking data to Lake Formation, crawling the data with Glue, and using Lake Formation for policies, the solution achieves the requirements with the least operational overhead.
Authoritative Links:

AWS Lake Formation: https://aws.amazon.com/lake-formation/
AWS Glue: https://aws.amazon.com/glue/

Question: 231

Exam Heist

A machine learning (ML) specialist is developing a deep learning sentiment analysis model that is based on data from movie reviews. After the ML specialist trains the model and reviews the model results on the validation set, the ML specialist discovers that the model is overfitting.

Which solutions will MOST improve the model generalization and reduce overfitting? (Choose three.)

- [ ] Shuffle the dataset with a different seed.
- [ ] Decrease the learning rate.
- [ ] Increase the number of layers in the network.
- [x] Add L1 regularization and L2 regularization. ✓
- [x] Add dropout. ✓
- [x] Decrease the number of layers in the network. ✓

Explanation:
The chosen options (D, E, F) are most effective in improving model generalization and reducing overfitting.

D. Add L1 regularization and L2 regularization: Regularization techniques penalize complex models by adding a penalty term to the loss function. L1 regularization (Lasso) encourages sparsity by driving some feature weights to zero, effectively performing feature selection and simplifying the model. L2 regularization (Ridge) shrinks the weights towards zero, preventing any single weight from becoming too large and dominating the model. Both help to prevent the model from memorizing the training data. https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l1-regularization, https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization
E. Add dropout: Dropout is a regularization technique that randomly deactivates some neurons during training. This forces the remaining neurons to learn more robust features that are not dependent on specific neurons. By preventing neurons from co-adapting, dropout reduces overfitting and improves the model's ability to generalize to unseen data. https://jmlr.org/papers/v15/srivastava14a.html
F. Decrease the number of layers in the network: A smaller, less complex network has fewer parameters and is less prone to overfitting than a deep and complex network. By reducing the network's capacity, the model is less likely to memorize the training data and will focus on learning the underlying patterns. This simplifies the model and encourages generalization.
Options A, B, and C are incorrect:

A. Shuffling the dataset with a different seed only changes the order of examples and won't inherently reduce overfitting. It can help ensure proper distribution during training, but it's not a primary method to combat overfitting.
B. Decreasing the learning rate can help the model converge more smoothly, but it does not directly address overfitting. While sometimes helpful in fine-tuning, it's not a principal method for improving generalization.
C. Increasing the number of layers in the network increases the model complexity, exacerbating overfitting. Deeper networks tend to memorize training data unless regularization techniques are applied.

Question: 232

Exam Heist

An online advertising company is developing a linear model to predict the bid price of advertisements in real time with low-latency predictions. A data scientist has trained the linear model by using many features, but the model is overfitting the training dataset. The data scientist needs to prevent overfitting and must reduce the number of features.

Which solution will meet these requirements?

- Retrain the model with L1 regularization applied. ✓
- Retrain the model with L2 regularization applied.
- Retrain the model with dropout regularization applied.
- Retrain the model by using more data.

Explanation:
The best solution to prevent overfitting and reduce the number of features in a linear model for real-time bid price prediction, while maintaining low latency, is **A. Retrain the model with L1 regularization applied.**
Here's why:

Overfitting: The model is overfitting, meaning it's memorizing the training data instead of learning the underlying patterns, leading to poor performance on unseen data. Reducing the number of features or applying regularization techniques helps combat this.
L1 Regularization (Lasso): L1 regularization adds a penalty term to the model's cost function that is proportional to the *absolute value* of the coefficients. This penalty encourages the model to shrink some coefficients to exactly zero, effectively performing feature selection. Features with zero coefficients are removed from the model, reducing its complexity and preventing overfitting. This is directly relevant to the requirement of reducing the number of features.
L2 Regularization (Ridge): L2 regularization adds a penalty term to the cost function proportional to the *square* of the coefficients. While L2 regularization can also reduce overfitting by shrinking coefficients, it typically doesn't drive coefficients to zero. Instead, it makes them very small. Therefore, L2 regularization reduces the impact of less important features without completely eliminating them. This is not the optimal approach when the explicit goal is to reduce the number of features.
Dropout Regularization: Dropout is primarily used in neural networks. It randomly drops out (deactivates) some neurons during training, forcing the network to learn more robust features and preventing co-adaptation. Dropout isn't directly applicable to linear models.
More Data: While adding more data *can* sometimes help reduce overfitting, it isn't always feasible or cost-effective. Further, adding more data doesn't address the requirement of reducing the *number* of features.

Low Latency: L1 regularization inherently simplifies the model by removing features, leading to faster prediction times, which aligns with the low-latency requirement. A model with fewer features will naturally have lower latency.
In summary, L1 regularization provides a direct way to reduce overfitting *and* the number of features in a linear model, crucial for achieving the desired performance and low-latency requirements of real-time bid price prediction.
Supporting Links:

Regularization: https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l1-regularization
L1 vs. L2 Regularization: https://www.geeksforgeeks.org/difference-between-l1-and-l2-regularization/

Question: 233

Exam Heist

A credit card company wants to identify fraudulent transactions in real time. A data scientist builds a machine learning model for this purpose. The transactional data is captured and stored in Amazon S3. The historic data is already labeled with two classes: fraud (positive) and fair transactions (negative). The data scientist removes all the missing data and builds a classifier by using the XGBoost algorithm in Amazon SageMaker. The model produces the following results:

•True positive rate (TPR): 0.700
•False negative rate (FNR): 0.300
•True negative rate (TNR): 0.977
•False positive rate (FPR): 0.023
•Overall accuracy: 0.949

Which solution should the data scientist use to improve the performance of the model?

- Apply the Synthetic Minority Oversampling Technique (SMOTE) on the minority class in the training dataset. Retrain the model with the updated training data. ✓
- Apply the Synthetic Minority Oversampling Technique (SMOTE) on the majority class in the training dataset. Retrain the model with the updated training data.
- Undersample the minority class.
- Oversample the majority class.

Explanation:
The problem describes a machine learning model for fraud detection with imbalanced classes (fraudulent transactions are less frequent than legitimate ones). The evaluation metrics indicate a high overall accuracy (0.949), but the True Positive Rate (TPR) is only 0.700 and the False Negative Rate (FNR) is 0.300. This means the model is missing a significant portion of fraudulent transactions. This is typical in imbalanced datasets, where the model is biased towards the majority class (fair transactions) because it sees more examples of it during training.
Option A, applying SMOTE (Synthetic Minority Oversampling Technique) to the minority class (fraudulent transactions), is the most appropriate solution. SMOTE works by creating synthetic samples of the minority class. It selects instances that are close in feature space, draws a line between the instances, and creates a new data point at a point along that line. This helps to balance the class distribution without simply duplicating existing samples (which could lead to overfitting). Retraining the model with the SMOTE-augmented data gives the algorithm a more representative sample of the minority class, allowing it to learn the features of fraudulent transactions more effectively. The goal is to improve the TPR without significantly impacting the TNR.
Option B is incorrect because SMOTE should be applied to the minority class, not the majority class. Applying it to the majority class would further exacerbate the class imbalance. Options C and D address class imbalance but are less preferred. Undersampling the minority class (option C) removes genuine fraud cases from the training data, which can lead to the model learning even less about fraud patterns. Oversampling the majority class (option D) increases the number of fair transactions, further biasing the model towards the negative class and potentially exacerbating the issue.
SMOTE helps improve model performance in imbalanced classification tasks. Using SMOTE increases the model's ability to correctly identify fraud. By generating synthetic data points, SMOTE mitigates the impact of the skewed distribution, leading to a better-performing model, specifically in detecting positive (fraudulent) instances.
Further research:

SMOTE: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
Handling Imbalanced Datasets: https://developers.google.com/machine-learning/data-prep/transform/class-imbalance

Question: 234

Exam Heist

A company is training machine learning (ML) models on Amazon SageMaker by using 200 TB of data that is stored in Amazon S3 buckets. The training data consists of individual files that are each larger than 200 MB in size. The company needs a data access solution that offers the shortest

processing time and the least amount of setup.

Which solution will meet these requirements?

- Use File mode in SageMaker to copy the dataset from the S3 buckets to the ML instance storage.
- Create an Amazon FSx for Lustre file system. Link the file system to the S3 buckets.
- Create an Amazon Elastic File System (Amazon EFS) file system. Mount the file system to the training instances.
- Use FastFile mode in SageMaker to stream the files on demand from the S3 buckets. ✓

Explanation:

The best solution is D: Use FastFile mode in SageMaker to stream the files on demand from the S3 buckets. Here's why:

Large File Sizes & Data Volume: The dataset is 200 TB, and each file is > 200 MB. This indicates that a streaming or on-demand approach is more suitable than copying the entire dataset to the instance storage.
SageMaker FastFile Mode: FastFile mode is specifically designed for training jobs with large datasets stored in S3. It leverages parallel data loading and prefetching to efficiently stream data directly from S3 to the training instances. It avoids the upfront overhead of downloading the entire dataset, which is crucial given the 200 TB size.
S3 Optimization: FastFile mode is optimized for S3, taking advantage of S3's performance characteristics and scalability.
Reduced Setup: Using FastFile mode involves minimal setup, requiring only the configuration of the training job to utilize this mode.
File Mode Inefficiency (Option A): Copying 200 TB of data to instance storage (File mode) would take a significant amount of time and require instances with very large storage volumes, resulting in increased costs and longer training times.
FSx for Lustre Overhead (Option B): While FSx for Lustre offers high-performance storage, it's an over-engineered solution for this scenario. Creating and managing an FSx for Lustre file system introduces complexity and cost that are not necessary when SageMaker's FastFile mode is readily available. Furthermore, the initial data loading from S3 to FSx introduces overhead.
Amazon EFS Limitations (Option C): Amazon EFS is generally better suited for smaller files and general-purpose file sharing, and the latency of accessing data from EFS would significantly increase training time compared to directly accessing S3 through FastFile mode.
In summary: FastFile mode in SageMaker provides the best balance of performance, cost-effectiveness, and ease of setup for accessing a large dataset from S3 during training. It avoids the pitfalls of copying the entire dataset while providing optimized data streaming from S3.
References:

SageMaker Data Input Methods (Focus on "FastFile mode")
Amazon S3

Question: 235

Exam Heist

An online store is predicting future book sales by using a linear regression model that is based on past sales data. The data includes duration, a numerical feature that represents the number of days that a book has been listed in the online store. A data scientist performs an exploratory data analysis and discovers that the relationship between book sales and duration is skewed and non-linear.

Which data transformation step should the data scientist take to improve the predictions of the model?

- One-hot encoding
- Cartesian product transformation
- Quantile binning ✓
- Normalization

Explanation:

Here's a detailed justification for why Quantile Binning (option C) is the most suitable data transformation step in this scenario: The problem states that the relationship between book sales and the 'duration' feature is skewed and non-linear. Linear regression models work best when the relationship between the independent and dependent variables is linear. Skewed data and non-linear relationships can significantly degrade the performance of linear regression.

Quantile binning transforms a continuous numerical feature into discrete bins based on quantiles (e.g., quartiles, deciles). This helps to address non-linearity by grouping similar values together and allowing the model to learn different coefficients for each bin. By grouping durations into bins (e.g., "0-30 days," "31-60 days," etc.), the model can learn a different sales impact for each duration range, even if the underlying relationship isn't perfectly linear. This is preferable to fitting a single linear relationship to the entire skewed duration range.

Why other options are less suitable:

One-hot encoding: This is primarily used for categorical features, not continuous numerical features like 'duration'. Applying one-hot encoding directly to 'duration' without binning would create a large number of sparse features and not address the non-linearity effectively.
Cartesian product transformation: This creates interaction terms between features. While interactions can sometimes improve model performance, it doesn't address the underlying issue of the skewed and non-linear relationship between 'duration' and sales. It also increases the dimensionality of the data, potentially leading to overfitting.
Normalization: This scales numerical features to a similar range (e.g., 0 to 1). While normalization can be helpful for gradient descent-based algorithms, it doesn't directly address non-linearity. It preserves the shape of the data distribution, including the skewness.
Therefore, quantile binning is the most appropriate choice because it discretizes the continuous 'duration' feature, mitigating the effect of the non-linear and skewed relationship and allowing the linear regression model to better capture the underlying patterns in the data. It simplifies the relationship by grouping similar duration values, which is particularly beneficial when dealing with skewed distributions.
Authoritative Links:

Feature Engineering Techniques:https://developers.google.com/machine-learning/data-prep/transform/bucketing
Scikit-learn's KBinsDiscretizer (Related to Binning):https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html

Question: 236

Exam Heist

A company's data engineer wants to use Amazon S3 to share datasets with data scientists. The data scientists work in three departments: Finance. Marketing, and Human Resources. Each department has its own IAM user group. Some datasets contain sensitive information and should be accessed only by the data scientists from the Finance department.

How can the data engineer set up access to meet these requirements?

- Create an S3 bucket for each dataset. Create an ACL for each S3 bucket. For each S3 bucket that contains a sensitive dataset, set the ACL to allow access only from the Finance department user group. Allow all three department user groups to access each S3 bucket that contains a non-sensitive dataset.
- Create an S3 bucket for each dataset. For each S3 bucket that contains a sensitive dataset, set the bucket policy to allow access only from the Finance department user group. Allow all three department user groups to access each S3 bucket that contains a non-sensitive dataset.
- Create a single S3 bucket that includes two folders to separate the sensitive datasets from the non-sensitive datasets. For the Finance department user group, attach an IAM policy that provides access to both folders. For the Marketing and Human Resources department user groups, attach an IAM policy that provides access to only the folder that contains the non-sensitive datasets. ✓
- Create a single S3 bucket that includes two folders to separate the sensitive datasets from the non-sensitive datasets. Set the policy for the S3 bucket to allow only the Finance department user group to access the folder that contains the sensitive datasets. Allow all three department user groups to access the folder that contains the non-sensitive datasets.

Explanation:
Here's a detailed justification for why option C is the best solution for managing access to sensitive and non-sensitive datasets in S3 for different data science departments:
The key requirements are:
Share datasets using S3.
Different IAM user groups for Finance, Marketing, and Human Resources.
Restrict access to sensitive datasets to only the Finance department.
All departments should access non-sensitive datasets.

Option C suggests a single S3 bucket with two folders: one for sensitive data and one for non-sensitive data. This approach is efficient from a storage management perspective compared to creating a separate bucket for each dataset (as in options A and B). Using folders within a single bucket simplifies organization and cost management.
IAM policies are used to control access. Attaching an IAM policy to the Finance department's user group allows them access to both folders (sensitive and non-sensitive data). This fulfills the requirement that Finance data scientists have access to everything.
IAM policies attached to the Marketing and Human Resources department's user groups grant access *only* to the folder containing non-sensitive datasets. This effectively restricts them from accessing the sensitive data.
IAM policies are preferred over ACLs (Access Control Lists) for access control in AWS for several reasons, mainly for their more granular control and central management capabilities. S3 bucket policies can also be used but applying policies to IAM user groups enables easier management of user access at scale and better compliance.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-iam-id-based-policy-examples.html

Option D's usage of Bucket policy to restrict access at folder level might work but would need to incorporate both allow and deny operations and is therefore more complex to manage and reason with when access control is modified. Attaching permissions at an IAM role is typically the best approach in this scenario for simplicity, manageability, and scalability Options A and B, which propose creating a separate bucket per dataset and using ACLs (Option A) or bucket policies (Option B), are less efficient for dataset management and access control. Managing numerous buckets can be operationally complex. ACLs are generally less preferred for complex access control scenarios than IAM policies.Therefore, option C provides a balance between security, efficiency, and manageability, making it the optimal solution.

Question: 237

Exam Heist

A company operates an amusement park. The company wants to collect, monitor, and store real-time traffic data at several park entrances by using strategically placed cameras. The company's security team must be able to immediately access the data for viewing. Stored data must be indexed and must be accessible to the company's data science team.

Which solution will meet these requirements MOST cost-effectively?

- Use Amazon Kinesis Video Streams to ingest, index, and store the data. Use the built-in integration with Amazon Rekognition for viewing by the security team.
- Use Amazon Kinesis Video Streams to ingest, index, and store the data. Use the built-in HTTP live streaming (HLS) capability for viewing by the security team. ✓
- Use Amazon Rekognition Video and the GStreamer plugin to ingest the data for viewing by the security team. Use Amazon Kinesis Data Streams to index and store the data.
- Use Amazon Kinesis Data Firehose to ingest, index, and store the data. Use the built-in HTTP live streaming (HLS) capability for viewing by the security team.

Explanation:
Here's a detailed justification for why option B is the most cost-effective solution:
Option B: Use Amazon Kinesis Video Streams with HLS

Ingestion, Indexing, and Storage: Amazon Kinesis Video Streams is specifically designed for ingesting, storing, and indexing real-time video data from sources like cameras. It allows for low-latency ingestion, making it ideal for immediate security access. Kinesis Video Streams automatically indexes the data based on timestamps, facilitating efficient retrieval and analysis by the data science team.
Immediate Access for Security Team (HLS): Kinesis Video Streams offers built-in HTTP Live Streaming (HLS) capability. HLS allows the security team to view the live video streams using standard web browsers or video players, enabling immediate access without the need for specialized software or integration.
Cost-Effectiveness: Kinesis Video Streams is optimized for video data. HLS doesn't introduce additional services or costs.
Why other options are less suitable:

Option A (Rekognition for Viewing): While Amazon Rekognition can analyze video, using it *solely* for viewing is an expensive misuse. Rekognition incurs costs for processing and analyzing the video stream. Viewing the stream should not incur additional processing overhead. This makes the solution less cost-effective.
Option C (Rekognition Video and Kinesis Data Streams): This option misuses Rekognition as an ingestion mechanism, and Kinesis Data Streams isn't optimized for storing large video files. Kinesis Data Streams is better suited for data records rather than raw video. Ingesting the data is Rekognition is expensive.
Option D (Kinesis Data Firehose): Kinesis Data Firehose is designed to load streaming data into data lakes and data warehouses. While it supports data transformation, it is not ideal for low-latency video streaming and viewing. HLS capability doesn't imply that Firehose is ideal for this use case.
In summary, Option B leverages the strengths of Amazon Kinesis Video Streams for video ingestion, storage, and indexing, and utilizes the built-in HLS capability for cost-effective, immediate video viewing by the security team. Other options introduce unnecessary complexity or incur additional costs for features not directly required for the core requirements.
Supporting Links:

Amazon Kinesis Video Streams: https://aws.amazon.com/kinesis/video-streams/
HTTP Live Streaming (HLS): https://developer.apple.com/streaming/

Question: 238

Exam Heist

An engraving company wants to automate its quality control process for plaques. The company performs the process before mailing each customized plaque to a customer. The company has created an Amazon S3 bucket that contains images of defects that should cause a plaque to be

rejected. Low-confidence predictions must be sent to an internal team of reviewers who are using Amazon Augmented AI (Amazon A2I).

Which solution will meet these requirements?

- Use Amazon Textract for automatic processing. Use Amazon A2I with Amazon Mechanical Turk for manual review.
- Use Amazon Rekognition for automatic processing. Use Amazon A2I with a private workforce option for manual review. ✓
- Use Amazon Transcribe for automatic processing. Use Amazon A2I with a private workforce option for manual review.
- Use AWS Panorama for automatic processing. Use Amazon A2I with Amazon Mechanical Turk for manual review.

Explanation:
Here's a detailed justification for why option B is the most suitable solution:
The core requirement is to automate quality control of engraved plaques using images and route low-confidence predictions to internal reviewers.

Amazon Rekognition: This AWS service is designed for image and video analysis. It can identify objects, people, text, scenes, and activities. In this scenario, Rekognition's image analysis capabilities can be leveraged to detect defects in the plaque images stored in S3. https://aws.amazon.com/rekognition/
Amazon A2I with a private workforce: Amazon Augmented AI (A2I) is used for human review of machine learning predictions. The question specifies that low-confidence predictions need to be sent to an *internal* team of reviewers. This eliminates the Amazon Mechanical Turk option, as it uses external, crowd-sourced reviewers. The private workforce option in A2I allows you to route reviews to your own employees. https://aws.amazon.com/augmented-ai/
Why other options are incorrect:

A. Amazon Textract: This service is for extracting text from documents. It is not suitable for identifying visual defects in images.
C. Amazon Transcribe: This service is for converting speech to text. It is not relevant for image-based quality control.
D. AWS Panorama: While Panorama can be used for computer vision at the edge, it's not the most appropriate choice here. This scenario focuses on processing images already stored in S3 rather than a real-time edge application. Using Panorama would require additional hardware and a different architecture. Also, utilizing Amazon Mechanical Turk as workforce option doesn't align with the need for internal reviewers.
Therefore, the combination of Amazon Rekognition for automatic image analysis and Amazon A2I with a private workforce for human review of low-confidence predictions provides the most effective and appropriate solution for the engraving company's quality control process. Option B addresses the core requirements of automating the quality checks and leveraging the internal review team.

Question: 239

Exam Heist

A machine learning (ML) engineer at a bank is building a data ingestion solution to provide transaction features to financial ML models. Raw transactional data is available in an Amazon Kinesis data stream.

The solution must compute rolling averages of the ingested data from the data stream and must store the results in Amazon SageMaker Feature Store. The solution also must serve the results to the models in near real time.

Which solution will meet these requirements?

- Load the data into an Amazon S3 bucket by using Amazon Kinesis Data Firehose. Use a SageMaker Processing job to aggregate the data and to load the results into SageMaker Feature Store as an online feature group.
- Write the data directly from the data stream into SageMaker Feature Store as an online feature group. Calculate the rolling averages in place within SageMaker Feature Store by using the SageMaker GetRecord API operation.
- Consume the data stream by using an Amazon Kinesis Data Analytics SQL application that calculates the rolling averages. Generate a result stream. Consume the result stream by using a custom AWS Lambda function that publishes the results to SageMaker Feature Store as an online feature group. ✓
- Load the data into an Amazon S3 bucket by using Amazon Kinesis Data Firehose. Use a SageMaker Processing job to load the data into SageMaker Feature Store as an offline feature group. Compute the rolling averages at query time.

Explanation:
The correct answer is C because it provides a near real-time solution for computing rolling averages and storing the results in SageMaker Feature Store. Let's break down why the other options are less suitable and why option C is optimal:

Option A: Using Kinesis Data Firehose to load data into S3 and then using a SageMaker Processing job adds latency. SageMaker Processing jobs are typically used for batch processing, not near real-time computations.
https://aws.amazon.com/sagemaker/processing/

Option B: SageMaker Feature Store is primarily a storage and retrieval system. While it offers the `GetRecord` API, it doesn't inherently perform real-time aggregations like rolling averages on incoming data. Calculating rolling averages using `GetRecord` operations would be highly inefficient and not suitable for near real-time serving. https://aws.amazon.com/sagemaker/feature-store/

Option D: Using Kinesis Data Firehose and a SageMaker Processing job to load the data into an *offline* feature group in SageMaker Feature Store is designed for historical analysis and batch model training. Computing rolling averages at query time would lead to significant latency and is impractical for near real-time model serving. Offline feature groups are also not designed for low-latency access.

Justification for Option C:

Real-time Processing: Kinesis Data Analytics is designed for real-time processing of streaming data. Its SQL capabilities enable the computation of rolling averages using windowing functions. https://aws.amazon.com/kinesis/data-analytics/

Low Latency: Kinesis Data Analytics generates a result stream with the aggregated data, minimizing the delay between data ingestion and processing.

Integration with SageMaker Feature Store: A Lambda function efficiently consumes the result stream and publishes the rolling averages to SageMaker Feature Store as an *online* feature group. Lambda provides a serverless and scalable way to ingest the pre-computed features. https://aws.amazon.com/lambda/

Near Real-Time Serving: SageMaker Feature Store's online feature store is optimized for low-latency retrieval, enabling models to access the rolling averages in near real-time.

Scalability: Kinesis Data Analytics and Lambda are both highly scalable AWS services, making the solution capable of handling varying data volumes.

In summary, option C offers the most efficient and scalable solution for computing rolling averages from a Kinesis data stream and storing the results in SageMaker Feature Store for near real-time model serving, leveraging the strengths of Kinesis Data Analytics, Lambda, and SageMaker Feature Store's online feature group capabilities.

Question: 240

Exam Heist

Each morning, a data scientist at a rental car company creates insights about the previous day's rental car reservation demands. The company needs to automate this process by streaming the data to Amazon S3 in near real time. The solution must detect high-demand rental cars at each of the company's locations. The solution also must create a visualization dashboard that automatically refreshes with the most recent data.

Which solution will meet these requirements with the LEAST development time?

- Use Amazon Kinesis Data Firehose to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using Amazon QuickSight ML Insights. Visualize the data in QuickSight. ✓
- Use Amazon Kinesis Data Streams to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using the Random Cut Forest (RCF) trained model in Amazon SageMaker. Visualize the data in Amazon QuickSight.
- Use Amazon Kinesis Data Firehose to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using the Random Cut Forest (RCF) trained model in Amazon SageMaker. Visualize the data in Amazon QuickSight.
- Use Amazon Kinesis Data Streams to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using Amazon QuickSight ML Insights. Visualize the data in QuickSight.

Explanation:

The best solution is A because it leverages managed services that require minimal development time and effort.

Kinesis Data Firehose is designed for simple, direct streaming of data to destinations like S3. It simplifies data ingestion compared to Kinesis Data Streams, which requires more complex processing for data persistence.

Amazon QuickSight ML Insights provides automated machine learning anomaly detection without needing to train and manage a dedicated ML model in SageMaker. This drastically reduces the development overhead.

QuickSight offers native integration for visualizing data stored in S3 and supports automatic refreshing of dashboards, fulfilling the visualization requirement.

Option B and C involve SageMaker's Random Cut Forest (RCF). While RCF is capable of detecting outliers, utilizing it directly requires more setup, training, and management effort compared to QuickSight ML Insights.

Option D uses Kinesis Data Streams, which is more complex than Kinesis Data Firehose for simply streaming data to S3, adding unnecessary development time.

Therefore, by using Kinesis Data Firehose for streaming, QuickSight ML Insights for anomaly detection, and QuickSight for visualization, option A achieves the required functionality with the least development time and operational overhead.

https://aws.amazon.com/kinesis/data-firehose/https://aws.amazon.com/quicksight/ml-insights/

Question: 241

Exam Heist

A company operates an amusement park. The company wants to collect, monitor, and store real-time traffic data at several park entrances by using strategically placed cameras. The company's security team must be able to immediately access the data for viewing. Stored data must be indexed and must be accessible to the company's data science team.

Which solution will meet these requirements MOST cost-effectively?

- Use Amazon Kinesis Video Streams to ingest, index, and store the data. Use the built-in integration with Amazon Rekognition for viewing by the security team.
- Use Amazon Kinesis Video Streams to ingest, index, and store the data. Use the built-in HTTP live streaming (HLS) capability for viewing by the security team. ✓
- Use Amazon Rekognition Video and the GStreamer plugin to ingest the data for viewing by the security team. Use Amazon Kinesis Data Streams to index and store the data.
- Use Amazon Kinesis Data Firehose to ingest, index, and store the data. Use the built-in HTTP live streaming (HLS) capability for viewing by the security team.

Explanation:

Here's a detailed justification for why option B is the most cost-effective solution for the amusement park's requirements:
The problem necessitates real-time video ingestion, immediate access for the security team, indexed storage, and accessibility for the data science team. Amazon Kinesis Video Streams is specifically designed for securely streaming video from devices to AWS. It provides built-in features for indexing and storing video data, which addresses the storage and accessibility requirements for the data science team.
Option B leverages Kinesis Video Streams' HTTP Live Streaming (HLS) capability. HLS allows the security team to view the live video streams immediately using standard video players, offering low latency and ease of access without needing specialized integrations. This is a cost-effective approach since it avoids the need for additional services or complex configurations for real-time viewing.
Option A suggests using Amazon Rekognition integration for viewing. While Rekognition is powerful for video analytics, it's not primarily designed for live video viewing by security personnel. It's also a more expensive service because of its object recognition capabilities.
Option C suggests using Amazon Rekognition Video *and* Kinesis Data Streams. This is more complex and therefore more costly than using Kinesis Video Streams for the entire process. Rekognition is overkill if the immediate need is simply viewing the video feed, not conducting real-time analysis of the feed. GStreamer adds complexity for ingestion. Kinesis Data Streams is optimized for data stream processing, not necessarily storing and indexing video.
Option D proposes using Kinesis Data Firehose. Firehose is designed for loading streaming data into data lakes and data warehouses, *not* for real-time video viewing and indexing in the way that Kinesis Video Streams is. While it can store data, it doesn't offer the built-in HLS streaming capability for real-time access.
Therefore, Kinesis Video Streams with its HLS capability provides the most cost-effective and straightforward solution by addressing all requirements within a single service, minimizing the need for complex integrations or additional expensive services like Rekognition for the initial task of viewing the live video feed.
Referential links:

Amazon Kinesis Video Streams: https://aws.amazon.com/kinesis/video-streams/
HTTP Live Streaming (HLS): https://developer.apple.com/streaming/

Question: 242

Exam Heist

An engraving company wants to automate its quality control process for plaques. The company performs the process before mailing each customized plaque to a customer. The company has created an Amazon S3 bucket that contains images of defects that should cause a plaque to be rejected. Low-confidence predictions must be sent to an internal team of reviewers who are using Amazon Augmented AI (Amazon A2I).

Which solution will meet these requirements?

- Use Amazon Textract for automatic processing. Use Amazon A2I with Amazon Mechanical Turk for manual review.
- Use Amazon Rekognition for automatic processing. Use Amazon A2I with a private workforce option for manual review. ✓
- Use Amazon Transcribe for automatic processing. Use Amazon A2I with a private workforce option for manual review.
- Use AWS Panorama for automatic processing. Use Amazon A2I with Amazon Mechanical Turk for manual review.

Explanation:

Here's a detailed justification for why option B is the correct solution, along with supporting information:

The problem requires identifying defects in images of plaques, automating the process as much as possible, and routing low-confidence predictions to human reviewers.

Amazon Rekognition: Rekognition is a service designed for image and video analysis. It can perform object detection, image classification, and facial recognition, making it suitable for identifying defects in the plaque images. It can be used for automated quality checking. https://aws.amazon.com/rekognition/
Amazon Augmented AI (Amazon A2I): A2I allows you to integrate human review workflows into your machine learning applications. This precisely matches the requirement to send low-confidence predictions to an internal team. https://aws.amazon.com/augmented-ai/
Private Workforce: The question specifies an *internal* team of reviewers. Amazon A2I offers a private workforce option, enabling you to use your own employees for manual review tasks, which aligns with the given requirement.
Therefore, using Amazon Rekognition to automatically process images and then sending low-confidence predictions to an internal team through Amazon A2I with a private workforce is the correct solution.
Why the other options are incorrect:

Amazon Textract: Textract extracts text and data from scanned documents. It's not designed for image analysis of defects. https://aws.amazon.com/textract/
Amazon Transcribe: Transcribe converts speech to text. It is not used for image defect detection. https://aws.amazon.com/transcribe/
AWS Panorama: Panorama is used for computer vision at the edge using on-premise devices. It doesn't fit the requirement of using an S3 bucket for images or imply edge deployment in the problem description. https://aws.amazon.com/panorama/
Amazon Mechanical Turk: While Mechanical Turk is a human workforce, it's a *public* workforce, not a private/internal one as specified in the requirement. A2I can use Mechanical Turk, but given the internal reviewer requirement, the private workforce option is more appropriate.

Question: 243

Exam Heist

A machine learning (ML) engineer at a bank is building a data ingestion solution to provide transaction features to financial ML models. Raw transactional data is available in an Amazon Kinesis data stream.

The solution must compute rolling averages of the ingested data from the data stream and must store the results in Amazon SageMaker Feature Store. The solution also must serve the results to the models in near real time.

Which solution will meet these requirements?

- Load the data into an Amazon S3 bucket by using Amazon Kinesis Data Firehose. Use a SageMaker Processing job to aggregate the data and to load the results into SageMaker Feature Store as an online feature group.
- Write the data directly from the data stream into SageMaker Feature Store as an online feature group. Calculate the rolling averages in place within SageMaker Feature Store by using the SageMaker GetRecord API operation.
- Consume the data stream by using an Amazon Kinesis Data Analytics SQL application that calculates the rolling averages. Generate a result stream. Consume the result stream by using a custom AWS Lambda function that publishes the results to SageMaker Feature Store as an online feature group. ✓
- Load the data into an Amazon S3 bucket by using Amazon Kinesis Data Firehose. Use a SageMaker Processing job to load the data into SageMaker Feature Store as an offline feature group. Compute the rolling averages at query time.

Explanation:
The optimal solution involves leveraging Amazon Kinesis Data Analytics for real-time aggregation and a Lambda function for efficient data ingestion into SageMaker Feature Store. Option C is the best choice because it efficiently calculates rolling averages in real-time using Kinesis Data Analytics SQL application. Kinesis Data Analytics allows for stream processing with SQL queries, enabling the computation of rolling averages directly from the Kinesis data stream. A custom Lambda function then consumes the result stream from Kinesis Data Analytics and writes the processed data to SageMaker Feature Store as an online feature group. This approach satisfies the near real-time requirement and uses appropriate services for stream processing and feature storage.
Option A is suboptimal because S3 and SageMaker Processing jobs are not designed for near real-time stream processing. Loading data to S3 and then triggering a processing job introduces latency and is better suited for batch processing. Also, it increases operational overhead.Option B is incorrect because SageMaker Feature Store is not designed to perform real-time calculations or aggregations like rolling averages. The `GetRecord` API is for retrieving feature values, not transforming them. The feature store primarily focuses on storing and serving features, not processing them.Option D is less efficient because an offline feature store and query-time aggregation would not meet the near real-time requirement. It increases the query execution time.
For authoritative information, refer to the following:

Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
AWS Lambda: https://aws.amazon.com/lambda/
Amazon SageMaker Feature Store: https://aws.amazon.com/sagemaker/feature-store/

Question: 244

Exam Heist

Each morning, a data scientist at a rental car company creates insights about the previous day's rental car reservation demands. The company needs to automate this process by streaming the data to Amazon S3 in near real time. The solution must detect high-demand rental cars at each of the company's locations. The solution also must create a visualization dashboard that automatically refreshes with the most recent data.

Which solution will meet these requirements with the LEAST development time?

- Use Amazon Kinesis Data Firehose to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using Amazon QuickSight ML Insights. Visualize the data in QuickSight. ✓
- Use Amazon Kinesis Data Streams to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using the Random Cut Forest (RCF) trained model in Amazon SageMaker. Visualize the data in Amazon QuickSight.
- Use Amazon Kinesis Data Firehose to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using the Random Cut Forest (RCF) trained model in Amazon SageMaker. Visualize the data in Amazon QuickSight.
- Use Amazon Kinesis Data Streams to stream the reservation data directly to Amazon S3. Detect high-demand outliers by using Amazon QuickSight ML Insights. Visualize the data in QuickSight.

Explanation:
The correct answer is A because it offers the least development time while meeting the requirements. Kinesis Data Firehose simplifies streaming data to S3, eliminating the need for manual data ingestion and management required by Kinesis Data Streams. QuickSight ML Insights directly integrates with S3 and provides automated outlier detection, removing the need to train and deploy a SageMaker model, which is a more complex and time-consuming process. QuickSight also facilitates immediate visualization.
Option B is incorrect because Kinesis Data Streams requires more development effort to manage and process the streaming data compared to Kinesis Data Firehose. Training and deploying an RCF model in SageMaker also adds significant development time.
Option C is incorrect as using a SageMaker RCF model increases complexity and development time. QuickSight ML Insights provides a faster route to outlier detection.
Option D is incorrect because Kinesis Data Streams is more complex to implement than Kinesis Data Firehose.
In summary, utilizing Kinesis Data Firehose for streaming data to S3 and using QuickSight's built-in ML Insights feature minimizes development time while providing outlier detection and visualization capabilities.
Relevant links:

Amazon Kinesis Data Firehose
Amazon QuickSight ML Insights
Amazon SageMaker Random Cut Forest

Question: 245

Exam Heist

A machine learning (ML) engineer is integrating a production model with a customer metadata repository for real-time inference. The repository is hosted in Amazon SageMaker Feature Store. The engineer wants to retrieve only the latest version of the customer metadata record for a single customer at a time.

Which solution will meet these requirements?

- Use the SageMaker Feature Store BatchGetRecord API with the record identifier. Filter to find the latest record.
- Create an Amazon Athena query to retrieve the data from the feature table.
- Create an Amazon Athena query to retrieve the data from the feature table. Use the write_time value to find the latest record.
- Use the SageMaker Feature Store GetRecord API with the record identifier. ✓

Explanation:
The correct answer is D: Use the SageMaker Feature Store GetRecord API with the record identifier.
Here's a detailed justification:
The scenario requires retrieving the *latest* version of a *single* customer metadata record from SageMaker Feature Store for real-time inference. The key here is real-time access to the latest version.

Option D (GetRecord API) is ideal because: The `GetRecord` API is specifically designed for retrieving the latest version of a single record from the online store of the feature group based on the record identifier (customer ID in this case). It provides low-latency access, perfect for real-time inference. This API directly retrieves the most recent record without requiring any filtering or complex queries.
Option A (BatchGetRecord API) is not the best choice because: While `BatchGetRecord` API can retrieve multiple records, it is less optimized for retrieving just a single record's latest version. Also, it retrieves all versions of a record. The prompt clearly mentions fetching just the latest version is a requirement, so this API does not fit. It involves filtering to get the latest record, adding unnecessary complexity.
Option B and C (Athena query) are unsuitable because: Using Amazon Athena for accessing feature store data is more appropriate for offline processing, data analysis, and model training where you need to query large datasets. It's not optimized for real-time, low-latency retrieval of a single record. Although Option C introduces the idea of filtering based on `write_time`, it is an unnecessary complication when a simpler API is available. Athena queries introduce latency and overhead compared to direct API calls to the online store.Specifically, Amazon Athena can only retrieve features from the *offline store* feature group. The prompt suggests that features will be accessed for real-time inference, which can only be done via the online store.
In summary, the `GetRecord` API directly addresses the requirement of fetching the latest version of a single record for real-time inference with minimal latency and complexity. The other options involve more complex querying and are not optimized for real-time access to the latest version of a single record.
Further Research:

SageMaker Feature Store Documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/feature-store.html
SageMaker Feature Store API Reference:
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_FeatureStore.html
SageMaker Feature Store GetRecord API:
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_FeatureStore_GetRecord.html

Question: 246

Exam Heist

A company's data scientist has trained a new machine learning model that performs better on test data than the company's existing model performs in the production environment. The data scientist wants to replace the existing model that runs on an Amazon SageMaker endpoint in the production environment. However, the company is concerned that the new model might not work well on the production environment data.

The data scientist needs to perform A/B testing in the production environment to evaluate whether the new model performs well on production environment data.

Which combination of steps must the data scientist take to perform the A/B testing? (Choose two.)

- [x] Create a new endpoint configuration that includes a production variant for each of the two models. ✓
- [ ] Create a new endpoint configuration that includes two target variants that point to different endpoints.
- [ ] Deploy the new model to the existing endpoint.
- [ ] Update the existing endpoint to activate the new model.
- [x] Update the existing endpoint to use the new endpoint configuration. ✓

Explanation:
The correct answer is AE because it outlines the proper procedure for implementing A/B testing on a SageMaker endpoint. Here's why:
A. Create a new endpoint configuration that includes a production variant for each of the two models. This is essential for A/B testing. An endpoint configuration defines how traffic is routed to different models. By creating a new endpoint configuration, the data scientist can specify that traffic should be split between the existing model and the new model. Each model gets its own production variant within this configuration. Production variants allow you to specify how much traffic to route to each model (e.g., 50% to the existing model, 50% to the new model). This variant-based routing is fundamental to A/B testing in SageMaker.
E. Update the existing endpoint to use the new endpoint configuration. Once the new endpoint configuration is created, the existing SageMaker endpoint needs to be updated to use it. This update tells the endpoint to start routing traffic according to the newly defined variant weights in the new configuration. Without updating the endpoint, the A/B testing setup will not be active. The original endpoint configuration would continue to serve the previous model exclusively.
Why other options are incorrect:

B. Create a new endpoint configuration that includes two target variants that point to different endpoints. Target variants are not a standard SageMaker construct for A/B testing within a single endpoint. The goal is to have both models served by a single endpoint for efficient traffic management and comparison. Creating separate endpoints adds unnecessary complexity.

C. Deploy the new model to the existing endpoint. Simply deploying the new model to the *existing* endpoint, without specifying how to divide traffic, would replace the existing model entirely. This does not allow for controlled A/B testing.
D. Update the existing endpoint to activate the new model. Similar to C, updating the existing endpoint to activate the new model, without an endpoint configuration specifying traffic split, would result in the new model replacing the old model, not A/B testing.
In summary, A and E together ensure that the new model and the existing model are both active and receiving a portion of production traffic, as defined in the new endpoint configuration. This allows the data scientist to compare the models' performance in a live environment.
Supporting Documentation:

Amazon SageMaker Inference: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
Endpoint Configurations: https://docs.aws.amazon.com/sagemaker/latest/dg/inference-endpoints.html
UpdateEndpoint API: https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_UpdateEndpoint.html

Question: 247

Exam Heist

A data scientist is working on a forecast problem by using a dataset that consists of .csv files that are stored in Amazon S3. The files contain a timestamp variable in the following format:

March 1st, 2020, 08:14pm -

There is a hypothesis about seasonal differences in the dependent variable. This number could be higher or lower for weekdays because some days and hours present varying values, so the day of the week, month, or hour could be an important factor. As a result, the data scientist needs to transform the timestamp into weekdays, month, and day as three separate variables to conduct an analysis.

Which solution requires the LEAST operational overhead to create a new dataset with the added features?

- Create an Amazon EMR cluster. Develop PySpark code that can read the timestamp variable as a string, transform and create the new variables, and save the dataset as a new file in Amazon S3.
- Create a processing job in Amazon SageMaker. Develop Python code that can read the timestamp variable as a string, transform and create the new variables, and save the dataset as a new file in Amazon S3.
- Create a new flow in Amazon SageMaker Data Wrangler. Import the S3 file, use the Featurize date/time transform to generate the new variables, and save the dataset as a new file in Amazon S3. ✓
- Create an AWS Glue job. Develop code that can read the timestamp variable as a string, transform and create the new variables, and save the dataset as a new file in Amazon S3.

Explanation:
The correct answer is C because Amazon SageMaker Data Wrangler is specifically designed for data preparation and feature engineering tasks with minimal coding. It offers a visual interface with pre-built transformations, including the "Featurize date/time" transform, which can directly extract weekdays, months, and days from a timestamp column with minimal coding effort. This reduces the operational overhead significantly compared to other options.
Option A (Amazon EMR) involves setting up and managing a cluster, writing PySpark code, and dealing with complexities of distributed data processing. Option B (SageMaker Processing Job) requires writing Python code and managing the execution environment, which is more overhead than Data Wrangler. Option D (AWS Glue job) necessitates writing code, defining schemas, and managing job execution, making it comparatively more complex and time-consuming.
Data Wrangler's drag-and-drop interface and pre-built transformations make it the fastest and easiest way to perform the required feature engineering with the least operational overhead. Other options require more code and infrastructure management. Data Wrangler streamlines the process, allowing the data scientist to focus on the analysis rather than infrastructure concerns.
Further research:

Amazon SageMaker Data Wrangler
SageMaker Data Wrangler Documentation

Question: 248

Exam Heist

A manufacturing company has a production line with sensors that collect hundreds of quality metrics. The company has stored sensor data and manual inspection results in a data lake for several months. To automate quality control, the machine learning team must build an automated mechanism that determines whether the produced goods are good quality, replacement market quality, or scrap quality based on the manual inspection results.

Which modeling approach will deliver the MOST accurate prediction of product quality?

- Amazon SageMaker DeepAR forecasting algorithm
- Amazon SageMaker XGBoost algorithm ✓
- Amazon SageMaker Latent Dirichlet Allocation (LDA) algorithm
- A convolutional neural network (CNN) and ResNet

Explanation:

The task is to classify manufactured goods into three categories (good, replacement, or scrap) based on sensor data. This is a multiclass classification problem.

Option B, Amazon SageMaker XGBoost algorithm, is the most suitable choice for several reasons. XGBoost is a powerful gradient boosting algorithm known for its high accuracy and ability to handle complex relationships within data. It excels in tabular data scenarios, which is the likely format of the sensor data and manual inspection results. It can also handle missing values. XGBoost is known for its regularization techniques, which help prevent overfitting, ensuring that the model generalizes well to new, unseen data from the production line. XGBoost implementations in SageMaker are highly optimized for performance and scalability.

Option A, Amazon SageMaker DeepAR forecasting algorithm, is designed for time-series forecasting, predicting future values based on historical trends. This is not relevant for classifying product quality based on current sensor readings.

Option C, Amazon SageMaker Latent Dirichlet Allocation (LDA) algorithm, is a topic modeling technique used for discovering abstract "topics" within a collection of documents. This is unsuitable for classification using sensor data.

Option D, a convolutional neural network (CNN) and ResNet, are typically used for image recognition tasks. While CNNs can be adapted to other data types, they are less efficient and more complex to implement than XGBoost for this tabular sensor data problem. CNNs require feature engineering to extract relevant features from the sensor data before training, which is a time consuming and expensive process. In addition, they require much more data than tree-based methods.

Therefore, XGBoost is the best option because it is optimized for the task and can deliver the most accurate predictions with the minimum effort.

Further research:

Amazon SageMaker XGBoost: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html
XGBoost Documentation: https://xgboost.readthedocs.io/en/stable/

Question: 249

Exam Heist

A healthcare company wants to create a machine learning (ML) model to predict patient outcomes. A data science team developed an ML model by using a custom ML library. The company wants to use Amazon SageMaker to train this model. The data science team creates a custom SageMaker image to train the model. When the team tries to launch the custom image in SageMaker Studio, the data scientists encounter an error within the application.

Which service can the data scientists use to access the logs for this error?

- Amazon S3
- Amazon Elastic Block Store (Amazon EBS)
- AWS CloudTrail
- Amazon CloudWatch ✓

Explanation:

Here's a detailed justification for why Amazon CloudWatch is the correct answer in this scenario:

The problem describes an error encountered when launching a custom SageMaker image within SageMaker Studio. Debugging such errors requires access to logs generated during the application's execution.

Amazon CloudWatch is a monitoring and observability service designed to collect and track metrics, collect and monitor log files, and set alarms. When running applications within SageMaker (or on any AWS service), you can configure the application to send its logs to CloudWatch. In this case, the error encountered while launching the custom SageMaker image will likely be logged and accessible through CloudWatch Logs. This includes application logs generated by SageMaker Studio, the custom image itself, and any underlying processes.

Amazon S3 is primarily for object storage. While you could potentially write logs to S3, it's not the standard or most efficient way to access real-time application logs for debugging. S3 requires additional configurations and processes to view and analyze the log data, unlike CloudWatch which is built for real-time log monitoring.

Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with EC2 instances. EBS volumes are not directly used for logging application errors in the way CloudWatch is. While the application may be running on an EC2 instance using EBS for its root volume, accessing the logs directly from the EBS volume would be complex and less practical than using CloudWatch.
AWS CloudTrail tracks API calls made to AWS services. While CloudTrail can be helpful for auditing and security analysis, it primarily focuses on recording API actions (e.g., "CreateImage," "RunTrainingJob"). It won't contain the detailed application-level logs needed to diagnose why the custom SageMaker image failed to launch within SageMaker Studio. CloudTrail captures who did what and when, not the internal workings or errors generated by an application.
Therefore, CloudWatch is the most appropriate service for accessing the error logs produced when the custom SageMaker image failed to launch. It provides the visibility required to understand the root cause of the issue and troubleshoot the custom image configuration.
Authoritative Links:

Amazon CloudWatch: https://aws.amazon.com/cloudwatch/
Amazon SageMaker Debugging with CloudWatch Logs: https://docs.aws.amazon.com/sagemaker/latest/dg/logging-cloudwatch.html

Question: 250

Exam Heist

A data scientist wants to build a financial trading bot to automate investment decisions. The financial bot should recommend the quantity and price of an asset to buy or sell to maximize long-term profit. The data scientist will continuously stream financial transactions to the bot for training purposes. The data scientist must select the appropriate machine learning (ML) algorithm to develop the financial trading bot.

Which type of ML algorithm will meet these requirements?

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning ✓

Explanation:
The correct answer is D, Reinforcement Learning. Here's why:
Reinforcement learning (RL) is ideally suited for training an agent (the trading bot in this case) to make sequential decisions within an environment (the financial market) to maximize a cumulative reward (long-term profit). The bot learns through trial and error, interacting directly with the real-time stream of financial transactions. RL doesn't need labeled data like supervised learning; instead, it receives feedback (rewards or penalties) based on the outcomes of its actions.
Supervised learning (A) requires labeled data (e.g., historical buy/sell decisions with associated profit outcomes) to train a model. While historical data can be used to initialize the bot's knowledge, it is insufficient for adapting to continuously changing market conditions. The bot needs to actively learn. Unsupervised learning (B) focuses on discovering patterns in unlabeled data and is unsuitable for making explicit trading recommendations. Semi-supervised learning (C) combines both labeled and unlabeled data, which might be useful in specific scenarios, but the core problem of learning optimal actions in a dynamic environment is best addressed by RL.
In the context of building a financial trading bot, RL offers a clear advantage because it allows the bot to adapt to evolving market dynamics and make decisions based on maximizing future rewards, which aligns perfectly with the goal of maximizing long-term profit. The agent (bot) continuously learns from the environment, optimizing its trading strategy over time.
For further research:

Amazon SageMaker RL: https://aws.amazon.com/sagemaker/reinforcement-learning/
Reinforcement Learning on AWS: https://aws.amazon.com/blogs/machine-learning/build-reinforcement-learning-applications-with-amazon-sagemaker-rl/
OpenAI's Introduction to Reinforcement Learning: https://spinningup.openai.com/en/latest/ (though not specific to AWS, it's a valuable resource for understanding RL concepts).

Question: 251

Exam Heist

A manufacturing company wants to create a machine learning (ML) model to predict when equipment is likely to fail. A data science team already constructed a deep learning model by using TensorFlow and a custom Python script in a local environment. The company wants to use Amazon SageMaker to train the model.

Which TensorFlow estimator configuration will train the model MOST cost-effectively?

- Turn on SageMaker Training Compiler by adding compiler_config=TrainingCompilerConfig() as a parameter. Pass the script to the estimator in the call to the TensorFlow fit() method.
- Turn on SageMaker Training Compiler by adding compiler_config=TrainingCompilerConfig() as a parameter. Turn on managed spot training by setting the use_spot_instances parameter to True. Pass the script to the estimator in the call to the TensorFlow fit() method. ✓
- Adjust the training script to use distributed data parallelism. Specify appropriate values for the distribution parameter. Pass the script to the estimator in the call to the TensorFlow fit() method.
- Turn on SageMaker Training Compiler by adding compiler_config=TrainingCompilerConfig() as a parameter. Set the MaxWaitTimeInSeconds parameter to be equal to the MaxRuntimeInSeconds parameter. Pass the script to the estimator in the call to the TensorFlow fit() method.

Explanation:

Here's a detailed justification for why option B is the most cost-effective solution for training the TensorFlow model using Amazon SageMaker:

The primary goal is cost optimization. Using SageMaker Training Compiler and managed spot training achieves this most effectively compared to the other options.

SageMaker Training Compiler: This compiler optimizes deep learning models for faster training on SageMaker. By automatically identifying and applying hardware-specific optimizations, it reduces the training time and, consequently, the cost. This is done by adding `compiler_config=TrainingCompilerConfig()` as a parameter to the TensorFlow estimator.

Managed Spot Training: This feature leverages spare Amazon EC2 capacity, often available at significantly reduced prices (up to 90% off on-demand prices). While spot instances can be interrupted, SageMaker handles this gracefully. When a spot instance is terminated, SageMaker automatically restarts the training job on another available spot instance (or on-demand if no spot instances are available within the specified time). Using `use_spot_instances=True` turns this feature on.

By combining these two methods, the training time is reduced due to optimized compilation, and the cost per unit of compute time is reduced using spot instances.

Why other options are less optimal:

Option A: Only using the SageMaker Training Compiler improves cost-effectiveness by reducing the training time. However, it doesn't leverage the potential cost savings offered by spot instances.

Option C: Distributed data parallelism can speed up training, but it can also introduce communication overhead, potentially requiring more expensive instances. It adds complexity and may not be as cost-effective if the model isn't perfectly suited for distributed training. The question mentions the model developed in a local environment with no initial indication of distribution.

Option D: Setting `MaxWaitTimeInSeconds` equal to `MaxRuntimeInSeconds` defeats the purpose of using spot instances. With spot training, it is possible that the instances are interrupted and recovered and may take longer overall, which the `MaxWaitTimeInSeconds` should accommodate. This defeats the purpose of managed spot training. Setting them equal is essentially forcing it to only use on-demand instances, negating the cost savings of spot instances.

In summary, Option B combines two powerful cost-saving techniques available within Amazon SageMaker, making it the most cost-effective solution for training the TensorFlow model.

Supporting Documentation:

Amazon SageMaker Training Compiler: https://docs.aws.amazon.com/sagemaker/latest/dg/training-compiler.html
Managed Spot Training: https://docs.aws.amazon.com/sagemaker/latest/dg/model-managed-spot-training.html

Question: 252

Exam Heist

An automotive company uses computer vision in its autonomous cars. The company trained its object detection models successfully by using transfer learning from a convolutional neural network (CNN). The company trained the models by using PyTorch through the Amazon SageMaker SDK.

The vehicles have limited hardware and compute power. The company wants to optimize the model to reduce memory, battery, and hardware consumption without a significant sacrifice in accuracy.

Which solution will improve the computational efficiency of the models?

- Use Amazon CloudWatch metrics to gain visibility into the SageMaker training weights, gradients, biases, and activation outputs. Compute the filter ranks based on the training information. Apply pruning to remove the low-ranking filters. Set new weights based on the pruned set of filters. Run a new training job with the pruned model.
- Use Amazon SageMaker Ground Truth to build and run data labeling workflows. Collect a larger labeled dataset with the labelling workflows. Run a new training job that uses the new labeled data with previous training data.
- Use Amazon SageMaker Debugger to gain visibility into the training weights, gradients, biases, and activation outputs. Compute the filter ranks based on the training information. Apply pruning to remove the low-ranking filters. Set the new weights based on the pruned set of filters. Run a new training job with the pruned model. ✓
- Use Amazon SageMaker Model Monitor to gain visibility into the ModelLatency metric and OverheadLatency metric of the model after the company deploys the model. Increase the model learning rate. Run a new training job.

Explanation:

The correct answer is C because it outlines a process for model pruning, a crucial technique for optimizing models for resource-constrained environments like autonomous vehicles.

Here's a detailed justification:

The problem requires optimizing a deep learning model (object detection CNN) for deployment on resource-limited devices. The primary goal is to reduce memory footprint and computational cost without significantly impacting accuracy. Model pruning is a proven technique to achieve this by removing less important weights or filters from the network.

Option C correctly employs **Amazon SageMaker Debugger** to gain visibility into the model's internals during training, specifically weights, gradients, biases, and activation outputs. This information is critical for determining the importance of different parts of the model. Debugger allows detailed insight into training tensors to identify candidate filters/neurons for pruning. [https://docs.aws.amazon.com/sagemaker/latest/dg/debugger.html]

Filter ranking is then performed based on the Debugger's output. Less important filters (those with low ranks based on their contribution to the model's performance) are identified and removed. This is the essence of pruning.

The remaining weights are then used to form a new, smaller model. Option C mentions setting new weights based on the pruned filter set, which implies fine-tuning or retraining the pruned model. A new training job, initiated with the pruned model, allows it to adapt and maintain accuracy with the reduced complexity.

Why other options are incorrect:

A: Using Amazon CloudWatch metrics is not suitable for gaining the granular visibility needed for model pruning. CloudWatch provides high-level operational data but not the detailed internal training information (weights, gradients) necessary for filter ranking.

B: Increasing the dataset size might improve overall accuracy but doesn't directly address the need to reduce the model's computational burden. More data generally leads to larger and more complex models initially. Moreover, Ground Truth focuses on data labeling, not model optimization.

D: Amazon SageMaker Model Monitor is for monitoring deployed models for data drift and concept drift *after* deployment, not for optimizing the model during training. Increasing the learning rate might speed up training but could also lead to instability and won't inherently reduce model size. Model monitor observes performance metrics like latency *post*-deployment. [https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html]

In summary, Option C leverages SageMaker Debugger to understand the inner workings of the model during training, applies pruning to remove less important parts, and fine-tunes the model to retain accuracy, which is the optimal approach for reducing the model size and computational requirements for deployment on autonomous vehicles with limited resources.

Question: 253

Exam Heist

A data scientist wants to improve the fit of a machine learning (ML) model that predicts house prices. The data scientist makes a first attempt to fit the model, but the fitted model has poor accuracy on both the training dataset and the test dataset.

Which steps must the data scientist take to improve model accuracy? (Choose three.)

- [ ] Increase the amount of regularization that the model uses.
- [x] Decrease the amount of regularization that the model uses. ✓
- [x] Increase the number of training examples that that model uses. ✓
- [ ] Increase the number of test examples that the model uses.
- [x] Increase the number of model features that the model uses. ✓
- [ ] Decrease the number of model features that the model uses.

Explanation:

The scenario describes a model that is underfitting, meaning it's not capturing the underlying patterns in the data. This is evidenced by poor accuracy on both the training and test datasets. To address underfitting, several strategies are applicable.
Increasing the number of training examples (C) is a crucial step. A larger training dataset provides the model with more information to learn from, potentially uncovering patterns that were previously missed. More data helps the model generalize better and improve its performance on unseen data.
Decreasing the amount of regularization (B) is important because excessive regularization can prevent the model from learning complex relationships within the data. Regularization techniques, like L1 or L2 regularization, are used to prevent overfitting by penalizing large coefficients. However, when underfitting, this penalty might be too strong, hindering the model's ability to fit the training data adequately.
Increasing the number of model features (E) can also help. Underfitting may occur when the model doesn't have enough features to represent the complexity of the data. Adding more relevant features can provide the model with a richer representation of the problem, allowing it to capture more intricate patterns and improve its predictive power.
Options A and F would typically address overfitting, which is the opposite problem. Increasing regularization (A) would further constrain the model and worsen the underfitting. Decreasing the number of features (F) could also lead to even poorer performance by reducing the model's capacity to learn. Option D, increasing the number of test examples, is about evaluating the model better, not improving the model itself. A larger test set can give a more reliable estimate of generalization error, but it does not improve the model's actual performance.
Therefore, the correct answer is BCE.
Relevant resources:

Overfitting and Underfitting With Machine Learning Algorithms: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/
Regularization: https://www.geeksforgeeks.org/regularization-in-machine-learning/

Question: 254

Exam Heist

A car company is developing a machine learning solution to detect whether a car is present in an image. The image dataset consists of one million images. Each image in the dataset is 200 pixels in height by 200 pixels in width. Each image is labeled as either having a car or not having a car.

Which architecture is MOST likely to produce a model that detects whether a car is present in an image with the highest accuracy?

- Use a deep convolutional neural network (CNN) classifier with the images as input. Include a linear output layer that outputs the probability that an image contains a car.
- Use a deep convolutional neural network (CNN) classifier with the images as input. Include a softmax output layer that outputs the probability that an image contains a car. ✓
- Use a deep multilayer perceptron (MLP) classifier with the images as input. Include a linear output layer that outputs the probability that an image contains a car.
- Use a deep multilayer perceptron (MLP) classifier with the images as input. Include a softmax output layer that outputs the probability that an image contains a car.

Explanation:
Here's a detailed justification for why option B is the most likely to produce the most accurate model for car detection in images, compared to the other options:
Justification:
Convolutional Neural Networks (CNNs) for Image Data: CNNs are specifically designed for processing image data. Their convolutional layers automatically learn spatial hierarchies of features. This is critical for image recognition tasks like car detection, where the location and arrangement of edges, shapes, and textures are important. MLPs, on the other hand, treat each pixel as an independent feature and don't inherently capture spatial relationships.
Deep Learning for Complex Features: The problem statement mentions a "deep" architecture. Deep learning models, especially deep CNNs, have multiple layers that enable them to learn complex, hierarchical representations of the input images. This is crucial for distinguishing between cars and non-car images, as cars can appear in varying conditions (lighting, angle, occlusion).
Softmax Output for Binary Classification: The problem is essentially a binary classification task: "car present" or "car absent." The softmax activation function is well-suited for the output layer in such cases. Softmax outputs a probability distribution over the classes, ensuring that the probabilities for all classes sum up to 1. In this case, it will output the probability of the image containing a car.
Linear vs. Softmax: While a linear output layer *could* be used and followed by a sigmoid function to produce a probability, softmax is a more natural and often better-performing choice for multi-class and binary classification problems when dealing with deep neural networks. Using softmax ensures that the output is a valid probability distribution.

Why Not MLP: MLPs would require flattening the 200x200 pixel image into a long vector, losing spatial information. This makes it significantly harder for the model to learn relevant features compared to CNNs. The number of parameters in a deep MLP would also be considerably larger, increasing the risk of overfitting, especially with a relatively complex dataset.

In summary, a deep CNN with a softmax output layer is the most appropriate architecture for this car detection task. The CNN leverages spatial information within the images, the deep architecture allows for learning complex features, and the softmax output ensures a well-calibrated probability for the presence of a car.**Authoritative Links for Further Research:**

Convolutional Neural Networks (CNNs): https://cs231n.github.io/convolutional-networks/ (Stanford CS231n course notes)
Softmax Function: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer
MLP vs CNN for Image Recognition: Research papers comparing MLP and CNN performance in image recognition tasks (search on Google Scholar).

Question: 255

Exam Heist

A company is creating an application to identify, count, and classify animal images that are uploaded to the company's website. The company is using the Amazon SageMaker image classification algorithm with an ImageNetV2 convolutional neural network (CNN). The solution works well for most animal images but does not recognize many animal species that are less common.

The company obtains 10,000 labeled images of less common animal species and stores the images in Amazon S3. A machine learning (ML) engineer needs to incorporate the images into the model by using Pipe mode in SageMaker.

Which combination of steps should the ML engineer take to train the model? (Choose two.)

- [ ] Use a ResNet model. Initiate full training mode by initializing the network with random weights.
- [ ] Use an Inception model that is available with the SageMaker image classification algorithm.
- [ ] Create a .lst file that contains a list of image files and corresponding class labels. Upload the .lst file to Amazon S3.
- [x] Initiate transfer learning. Train the model by using the images of less common species. ✓
- [x] Use an augmented manifest file in JSON Lines format. ✓

Explanation:
The correct answer is DE. Here's why:
D. Initiate transfer learning. Train the model by using the images of less common species.
Since the existing model (ImageNetV2 CNN) already performs well on common animal species, initiating transfer learning is the most efficient approach. Transfer learning leverages the pre-trained weights from the ImageNetV2 CNN model, which already contains valuable feature representations learned from a vast dataset of images. By training the model further on the 10,000 labeled images of less common animal species, the model can adapt its existing knowledge to recognize these new species without needing to be trained from scratch. This approach significantly reduces training time and resource consumption compared to training a model from random weights. https://aws.amazon.com/blogs/machine-learning/accelerate-deep-learning-with-transfer-learning-using-the-amazon-sagemaker-image-classification-algorithm/
E. Use an augmented manifest file in JSON Lines format.
Pipe mode is a SageMaker feature that streams data directly from S3 to the training instances. This eliminates the need to download the entire dataset before training, saving time and storage space. When using Pipe mode with image data, SageMaker requires the input data to be in a specific format. Augmented manifest files in JSON Lines format are the recommended way to provide the data location (S3 path of the images) and their corresponding labels to SageMaker for Pipe mode ingestion. This format allows SageMaker to efficiently stream and process the images and labels during training. https://docs.aws.amazon.com/sagemaker/latest/dg/augmented-manifest.html
Why the other options are less suitable:

A. Use a ResNet model. Initiate full training mode by initializing the network with random weights. While ResNet is a valid CNN architecture, starting from random weights would be very resource-intensive and time-consuming, especially when a pre-trained model like ImageNetV2 CNN already exists and performs well on other species.
B. Use an Inception model that is available with the SageMaker image classification algorithm. While Inception is another good model, it might not be significantly better than the already used ImageNetV2 CNN, and switching models would not address the core issue of a lack of training data for the less common species. Transfer learning is the more direct solution.
C. Create a .lst file that contains a list of image files and corresponding class labels. Upload the .lst file to Amazon S3. While .lst files are a valid input format for the SageMaker image classification algorithm, augmented manifest files offer more flexibility and are generally preferred, especially when working with Pipe mode, as it allows more efficient data streaming and integration with other AWS services. Augmented manifest files handle larger, more complex datasets more gracefully.

Question: 256

Exam Heist

A music streaming company is building a pipeline to extract features. The company wants to store the features for offline model training and online inference. The company wants to track feature history and to give the company's data science teams access to the features.

Which solution will meet these requirements with the MOST operational efficiency?

- Use Amazon SageMaker Feature Store to store features for model training and inference. Create an online store for online inference. Create an offline store for model training. Create an IAM role for data scientists to access and search through feature groups. ✓
- Use Amazon SageMaker Feature Store to store features for model training and inference. Create an online store for both online inference and model training. Create an IAM role for data scientists to access and search through feature groups.
- Create one Amazon S3 bucket to store online inference features. Create a second S3 bucket to store offline model training features. Turn on versioning for the S3 buckets and use tags to specify which tags are for online inference features and which are for offline model training features. Use Amazon Athena to query the S3 bucket for online inference. Connect the S3 bucket for offline model training to a SageMaker training job. Create an IAM policy that allows data scientists to access both buckets.
- Create two separate Amazon DynamoDB tables to store online inference features and offline model training features. Use time-based versioning on both tables. Query the DynamoDB table for online inference. Move the data from DynamoDB to Amazon S3 when a new SageMaker training job is launched. Create an IAM policy that allows data scientists to access both tables.

Explanation:

The most operationally efficient solution is A, leveraging Amazon SageMaker Feature Store. Here's why:

Centralized Feature Management: SageMaker Feature Store provides a centralized repository for managing features used for both online inference and offline training. This eliminates the need for separate storage solutions, reducing complexity.
Online and Offline Stores: Feature Store supports both an online store (low-latency access for real-time inference) and an offline store (for large-scale training). This duality caters to the specific requirements of each use case.
Feature History Tracking: Feature Store automatically tracks feature history, enabling reproducibility and debugging of models.
Data Scientist Access Control: IAM roles can be used to grant data scientists controlled access to feature groups, facilitating collaboration and experimentation while maintaining security.
Operational Efficiency: Managing separate S3 buckets (option C) or DynamoDB tables (option D) requires significant operational overhead for versioning, data movement, and querying. These options introduce unnecessary complexity compared to the managed Feature Store solution.
Scalability and Performance: Feature Store is designed for scalability and performance, handling the demands of both online inference and offline training scenarios.
SageMaker Integration: Feature Store is tightly integrated with SageMaker, simplifying model training and deployment workflows.
Option B is less ideal because using the online store for offline training is generally inefficient. Online stores prioritize low latency and may not be optimized for the large-scale data access required for model training.
Options C and D involve manually managing infrastructure and data pipelines, which is less efficient than using a managed service like Feature Store.
Authoritative Links:

Amazon SageMaker Feature Store: https://aws.amazon.com/sagemaker/feature-store/
Feature Store concepts: https://docs.aws.amazon.com/sagemaker/latest/dg/feature-store.html

Question: 257

Exam Heist

A beauty supply store wants to understand some characteristics of visitors to the store. The store has security video recordings from the past several years. The store wants to generate a report of hourly visitors from the recordings. The report should group visitors by hair style and hair color.

Which solution will meet these requirements with the LEAST amount of effort?

- Use an object detection algorithm to identify a visitor's hair in video frames. Pass the identified hair to an ResNet-50 algorithm to determine hair style and hair color. ✓
- Use an object detection algorithm to identify a visitor's hair in video frames. Pass the identified hair to an XGBoost algorithm to determine hair style and hair color.
- Use a semantic segmentation algorithm to identify a visitor's hair in video frames. Pass the identified hair to an ResNet-50 algorithm to determine hair style and hair color.
- Use a semantic segmentation algorithm to identify a visitor's hair in video frames. Pass the identified hair to an XGBoost algorithm to determine hair style and hair.

Explanation:
The correct answer is A because it offers the most efficient and appropriate solution for the given problem with the least amount of effort. Here's a detailed justification:
Object Detection vs. Semantic Segmentation: Object detection focuses on identifying and localizing distinct objects within an image or video frame. Semantic segmentation, on the other hand, aims to classify each pixel in an image, assigning it to a specific object category. Since the task is to identify and analyze a specific object (hair), object detection is more directly suited than segmentation, which would require more processing.
ResNet-50 for Image Classification: ResNet-50 is a convolutional neural network (CNN) known for its deep residual learning architecture. It is highly effective for image classification tasks, including determining hair style and color based on visual features. Transfer learning can be used to reduce the amount of training data and effort required.

https://arxiv.org/abs/1512.03385 (ResNet paper)
XGBoost's Suitability: XGBoost is a powerful gradient boosting algorithm, primarily used for structured/tabular data and classification/regression problems. While it can be adapted for image-related tasks, it is not directly applicable to image classification of hair styles and colors in this scenario.
Least Amount of Effort: The proposed solution with object detection and ResNet-50 is relatively straightforward. Object detection quickly locates the hair region, and the pre-trained ResNet-50 model can then be fine-tuned for classifying hair style and color. XGBoost usage would require significant feature extraction and data engineering to become effective, therefore requiring more effort.
Why other choices are not preferred: Option C, utilizing semantic segmentation, requires pixel-level classification, leading to a more computationally expensive and complex solution. B and D also suggest using XGBoost which isn't ideally suited for image classification.
Scalability and Automation: AWS services can be used to easily scale the system. Security recordings can be ingested by AWS S3 and processed by AWS SageMaker. The generated hourly visitor reports with the grouped hair styles and colors can be stored in AWS DynamoDB.

In summary, object detection to identify hair followed by a ResNet-50-based image classification system offers the simplest and most efficient approach for analyzing hair style and color from video recordings.

Question: 258

Exam Heist

A financial services company wants to automate its loan approval process by building a machine learning (ML) model. Each loan data point contains credit history from a third-party data source and demographic information about the customer. Each loan approval prediction must come with a report that contains an explanation for why the customer was approved for a loan or was denied for a loan. The company will use Amazon SageMaker to build the model.

Which solution will meet these requirements with the LEAST development effort?

- Use SageMaker Model Debugger to automatically debug the predictions, generate the explanation, and attach the explanation report.
- Use AWS Lambda to provide feature importance and partial dependence plots. Use the plots to generate and attach the explanation report.
- Use SageMaker Clarify to generate the explanation report. Attach the report to the predicted results. ✓
- Use custom Amazon CloudWatch metrics to generate the explanation report. Attach the report to the predicted results.

Explanation:
The correct answer is C: Use SageMaker Clarify to generate the explanation report. Attach the report to the predicted results. SageMaker Clarify is specifically designed to detect potential bias in machine learning models and explain their predictions. It provides feature importance scores, which can be used to understand why a model made a particular prediction. Crucially, it automates the generation of these explanations in a structured report. This aligns perfectly with the requirement of providing an explanation report along with each loan approval prediction. Using Clarify requires minimal custom coding as it is integrated into the SageMaker ecosystem.

Option A, SageMaker Model Debugger, primarily focuses on identifying and fixing training issues, such as vanishing gradients or overfitting. While debugging is useful, it doesn't directly generate the required explanation reports for individual predictions.
Option B, using AWS Lambda to generate feature importance and partial dependence plots, would require significantly more development effort. It would involve custom coding to calculate and visualize these metrics and then create a report based on the results. This approach lacks the automation and pre-built capabilities of SageMaker Clarify.
Option D, custom Amazon CloudWatch metrics, are primarily for monitoring the performance of the model and underlying infrastructure, not for explaining individual predictions. While useful for operational insights, CloudWatch metrics lack the analytical capabilities to understand individual loan approval decisions.
Therefore, SageMaker Clarify provides the most direct, efficient, and least effort-intensive solution for generating explanation reports for loan approval predictions within the SageMaker environment. It's a purpose-built tool that integrates seamlessly and reduces the burden of custom coding.
Supporting links:

SageMaker Clarify Documentation

Question: 259

Exam Heist

A financial company sends special offers to customers through weekly email campaigns. A bulk email marketing system takes the list of email addresses as an input and sends the marketing campaign messages in batches. Few customers use the offers from the campaign messages. The company does not want to send irrelevant offers to customers.

A machine learning (ML) team at the company is using Amazon SageMaker to build a model to recommend specific offers to each customer based on the customer's profile and the offers that the customer has accepted in the past.

Which solution will meet these requirements with the MOST operational efficiency?

- Use the Factorization Machines algorithm to build a model that can generate personalized offer recommendations for customers. Deploy a SageMaker endpoint to generate offer recommendations. Feed the offer recommendations into the bulk email marketing system.
- Use the Neural Collaborative Filtering algorithm to build a model that can generate personalized offer recommendations for customers. Deploy a SageMaker endpoint to generate offer recommendations. Feed the offer recommendations into the bulk email marketing system.
- Use the Neural Collaborative Filtering algorithm to build a model that can generate personalized offer recommendations for customers. Deploy a SageMaker batch inference job to generate offer recommendations. Feed the offer recommendations into the bulk email marketing system.
- Use the Factorization Machines algorithm to build a model that can generate personalized offer recommendations for customers. Deploy a SageMaker batch inference job to generate offer recommendations. Feed the offer recommendations into the bulk email marketing system. ✓

Explanation:
The most operationally efficient solution is **D**. Here's why:

Factorization Machines (FM) vs. Neural Collaborative Filtering (NCF): FMs are generally faster and more computationally efficient than NCF, especially when dealing with sparse data, which is common in recommendation systems. This leads to faster model training and inference, reducing operational costs. FM effectively models feature interactions, crucial for personalized recommendations.

Factorization Machines
Batch Inference vs. Real-time Endpoint: Batch inference is significantly more cost-effective and operationally simpler for weekly email campaigns. You only need to generate recommendations once a week, rather than maintaining a real-time endpoint that constantly consumes resources. Batch inference allows processing recommendations in bulk during off-peak hours, optimizing resource utilization.

SageMaker Batch Transform
Operational Efficiency: By using FMs with batch inference, you minimize infrastructure costs, reduce latency concerns, and simplify the overall architecture. The model trains and infers periodically instead of consuming continuous resources, making weekly email campaigns more efficient.
Scalability: Batch processing is designed for handling large datasets, making it scalable to accommodate growing customer bases.
Options A and B require maintaining a SageMaker endpoint, resulting in constant costs, and are less efficient given the weekly nature of the task. Option C uses Neural Collaborative Filtering, which is more computationally expensive compared to Factorization Machines.

Question: 260

Exam Heist

A social media company wants to develop a machine learning (ML) model to detect inappropriate or offensive content in images. The company has collected a large dataset of labeled images and plans to use the built-in Amazon SageMaker image classification algorithm to train the model. The company also intends to use SageMaker pipe mode to speed up the training.

The company splits the dataset into training, validation, and testing datasets. The company stores the training and validation images in folders that are named Training and Validation, respectively. The folders contain subfolders that correspond to the names of the dataset classes. The company resizes the images to the same size and generates two input manifest files named training.lst and validation.lst, for the training dataset and the validation dataset, respectively. Finally, the company creates two separate Amazon S3 buckets for uploads of the training dataset and the validation dataset.

Which additional data preparation steps should the company take before uploading the files to Amazon S3?

- Generate two Apache Parquet files, training.parquet and validation.parquet, by reading the images into a Pandas data frame and storing the data frame as a Parquet file. Upload the Parquet files to the training S3 bucket.
- Compress the training and validation directories by using the Snappy compression library. Upload the manifest and compressed files to the training S3 bucket.
- Compress the training and validation directories by using the gzip compression library. Upload the manifest and compressed files to the training S3 bucket.
- Generate two RecordIO files, training.rec and validation.rec, from the manifest files by using the im2rec Apache MXNet utility tool. Upload the RecordIO files to the training S3 bucket. ✓

Explanation:

The correct answer is **D. Generate two RecordIO files, training.rec and validation.rec, from the manifest files by using the im2rec Apache MXNet utility tool. Upload the RecordIO files to the training S3 bucket.**

Here's why:

Amazon SageMaker's built-in image classification algorithm, especially when used with Pipe mode for faster training, expects data to be in a specific format for efficient ingestion. RecordIO is a highly optimized binary data format designed to accelerate data transfer to the training instances. The `im2rec` tool, part of Apache MXNet, is specifically designed for converting image datasets into the RecordIO format from manifest files (like `training.lst` and `validation.lst`). These manifest files contain lists of images and their corresponding labels.

Pipe mode uses channels to stream data directly from S3 to the training instances, bypassing the need to download the entire dataset to the instance's local storage. RecordIO format, combined with Pipe mode, enables SageMaker to read data in parallel, thus significantly speeding up the training process. By converting the images to RecordIO format, the images are efficiently packaged and indexed for optimized input during the training job.

Option A is incorrect because while Parquet is a columnar storage format suitable for analytics, it's not the preferred format for SageMaker's image classification algorithm, especially when using Pipe mode. Option B and C suggest compression, but it doesn't address the required data format for the algorithm to process the data directly from S3 efficiently. While compression can save storage space, it doesn't prepare the data for efficient streaming through SageMaker Pipe mode. The image classification algorithm expects to be able to readily access individual images and their labels from the training dataset. Therefore, generating RecordIO files from the manifest files using `im2rec` is the necessary step to prepare the image data for efficient training with the SageMaker image classification algorithm using Pipe mode.

Relevant Documentation:

SageMaker Image Classification Algorithm Input/Output Interface
Using Pipe Input Mode for Training
Apache MXNet im2rec Tool

Question: 261

Exam Heist

A media company wants to create a solution that identifies celebrities in pictures that users upload. The company also wants to identify the IP address and the timestamp details from the users so the company can prevent users from uploading pictures from unauthorized locations.

Which solution will meet these requirements with LEAST development effort?

- Use AWS Panorama to identify celebrities in the pictures. Use AWS CloudTrail to capture IP address and timestamp details.
- Use AWS Panorama to identify celebrities in the pictures. Make calls to the AWS Panorama Device SDK to capture IP address and timestamp details.
- Use Amazon Rekognition to identify celebrities in the pictures. Use AWS CloudTrail to capture IP address and timestamp details. ✓
- Use Amazon Rekognition to identify celebrities in the pictures. Use the text detection feature to capture IP address and timestamp details.

Explanation:
The correct answer is C. Let's break down why:

Celebrity Recognition: Amazon Rekognition is a specialized service designed for image and video analysis, including facial recognition and celebrity identification. AWS Panorama, while capable of image analysis at the edge, is primarily focused on integrating computer vision into existing on-premise camera systems and industrial environments. Rekognition provides a ready-to-use, scalable solution for identifying celebrities in images uploaded by users.
IP Address and Timestamp Tracking: AWS CloudTrail is a service that logs API calls made to AWS services. When a user uploads an image, the associated API call to a service like Amazon S3 (where the image might be stored) or Rekognition itself will be recorded in CloudTrail. These logs include the source IP address of the request and the timestamp of the event. This allows tracking the origin and time of the uploads without requiring any custom code.
Why other options are incorrect:

A & B (AWS Panorama): Panorama is not designed for processing user-uploaded images directly in the cloud. It's for analyzing video streams from connected cameras. Therefore, it's not the right tool for identifying celebrities in uploaded pictures. Option B additionally mentions the Panorama Device SDK, which is specific to Panorama's edge computing environment.
D (Text Detection): While Rekognition does have a text detection feature, it would be highly unreliable and inefficient to extract IP addresses and timestamps from images of users uploading the image. CloudTrail directly provides this information. Therefore, the solution using Amazon Rekognition for celebrity identification and AWS CloudTrail for IP address and timestamp tracking offers the least development effort because it leverages managed AWS services specifically designed for these tasks. It avoids custom coding, ensures scalability, and follows best practices for security and auditing.
Supporting Links:

Amazon Rekognition: https://aws.amazon.com/rekognition/
AWS CloudTrail: https://aws.amazon.com/cloudtrail/
AWS Panorama: https://aws.amazon.com/panorama/

Question: 262

Exam Heist

A pharmaceutical company performs periodic audits of clinical trial sites to quickly resolve critical findings. The company stores audit documents in text format. Auditors have requested help from a data science team to quickly analyze the documents. The auditors need to discover the 10 main topics within the documents to prioritize and distribute the review work among the auditing team members. Documents that describe adverse events must receive the highest priority.

A data scientist will use statistical modeling to discover abstract topics and to provide a list of the top words for each category to help the auditors assess the relevance of the topic.

Which algorithms are best suited to this scenario? (Choose two.)

- [x] Latent Dirichlet allocation (LDA) ✓
- [ ] Random forest classifier
- [x] Neural topic modeling (NTM) ✓
- [ ] Linear support vector machine
- [ ] Linear regression

Explanation:
Here's a detailed justification for why Latent Dirichlet Allocation (LDA) and Neural Topic Modeling (NTM) are the best algorithms for the pharmaceutical company's document analysis scenario, along with why the other options are not well-suited:
Why LDA and NTM are suitable:

Topic Discovery: LDA and NTM are unsupervised machine learning techniques specifically designed for topic modeling. They excel at uncovering latent (hidden) topics within a collection of documents. The data scientist can use these algorithms to automatically identify the 10 main topics from the audit documents, as requested.
Abstract Topics and Top Words: Both algorithms output a set of topics, where each topic is represented by a probability distribution over words. This allows the data scientist to provide the auditors with a list of the top words associated with each topic. This is crucial for assessing the relevance of each topic to the audits and prioritizing documents describing adverse events.
Unsupervised Learning: The scenario doesn't mention pre-labeled data. LDA and NTM can work without labeled training data, fitting the requirement that the algorithm must discover topics itself.

Scalability: LDA and NTM can handle a large volume of audit documents, making them suitable for real-world applications. AWS provides infrastructure and services that can easily scale to handle large datasets.
Why the other options are unsuitable:

Random Forest Classifier (B): This is a supervised classification algorithm. It requires labeled training data, where documents are pre-classified into specific categories. This doesn't align with the goal of discovering topics without prior knowledge.
Linear Support Vector Machine (D): Similar to Random Forest, SVM is a supervised classification algorithm. It's not designed for topic modeling and requires labeled data.
Linear Regression (E): This algorithm is used for predicting a continuous target variable. It's not relevant to the task of discovering topics within text documents.
In summary: LDA and NTM are the most appropriate algorithms because they directly address the problem of topic discovery in unstructured text data. They are unsupervised, can identify abstract topics, and provide lists of top words to help auditors assess relevance and prioritize their work.
Authoritative Links:

Latent Dirichlet Allocation (LDA): https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
Neural Topic Modeling (NTM): Research papers and libraries (e.g., PyTorch, TensorFlow) are primary resources. You can find them on Google Scholar.

Question: 263

Exam Heist

A company needs to deploy a chatbot to answer common questions from customers. The chatbot must base its answers on company documentation.

Which solution will meet these requirements with the LEAST development effort?

- Index company documents by using Amazon Kendra. Integrate the chatbot with Amazon Kendra by using the Amazon Kendra Query API operation to answer customer questions. ✓
- Train a Bidirectional Attention Flow (BiDAF) network based on past customer questions and company documents. Deploy the model as a real-time Amazon SageMaker endpoint. Integrate the model with the chatbot by using the SageMaker Runtime InvokeEndpoint API operation to answer customer questions.
- Train an Amazon SageMaker Blazing Text model based on past customer questions and company documents. Deploy the model as a real-time SageMaker endpoint. Integrate the model with the chatbot by using the SageMaker Runtime InvokeEndpoint API operation to answer customer questions.
- Index company documents by using Amazon OpenSearch Service. Integrate the chatbot with OpenSearch Service by using the OpenSearch Service k-nearest neighbors (k-NN) Query API operation to answer customer questions.

Explanation:
The best solution is to use Amazon Kendra and its Query API (Option A). Here's why:

Amazon Kendra is a managed intelligent search service: Designed to understand natural language queries and return relevant information from documents. This directly addresses the need for a chatbot that can answer questions based on company documentation.
Least Development Effort: Kendra handles the complexities of indexing, searching, and understanding documents. You simply point it to your data sources, and it automatically builds an index. The Query API provides a simple way to integrate Kendra into your chatbot application.
No Machine Learning Expertise Required: Alternatives B and C require building and deploying custom machine learning models using SageMaker, which needs considerable time, effort, and expertise in model training, tuning, and deployment.
Managed Service: Kendra handles scaling, availability, and maintenance, reducing operational overhead. Options involving custom models demand managing these aspects.
Opensearch is not built to answer questions from text: Opensearch requires to use its KNN feature that needs additional development and has less integration with chatbots compared to Kendra.
Options B and C involve training custom machine learning models (BiDAF and BlazingText, respectively). While these could potentially work, they require significantly more development effort, ML expertise, and infrastructure management compared to using Amazon Kendra. Option D, while it can index documents, does not provide the same level of natural language understanding and question answering capabilities as Kendra out-of-the-box.
Supporting Links:

Amazon Kendra: https://aws.amazon.com/kendra/
Amazon Kendra Query API: https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html

Question: 264

Exam Heist

A company wants to conduct targeted marketing to sell solar panels to homeowners. The company wants to use machine learning (ML) technologies to identify which houses already have solar panels. The company has collected 8,000 satellite images as training data and will use Amazon SageMaker Ground Truth to label the data.

The company has a small internal team that is working on the project. The internal team has no ML expertise and no ML experience.

Which solution will meet these requirements with the LEAST amount of effort from the internal team?

- Set up a private workforce that consists of the internal team. Use the private workforce and the SageMaker Ground Truth active learning feature to label the data. Use Amazon Rekognition Custom Labels for model training and hosting. ✓
- Set up a private workforce that consists of the internal team. Use the private workforce to label the data. Use Amazon Rekognition Custom Labels for model training and hosting.
- Set up a private workforce that consists of the internal team. Use the private workforce and the SageMaker Ground Truth active learning feature to label the data. Use the SageMaker Object Detection algorithm to train a model. Use SageMaker batch transform for inference.
- Set up a public workforce. Use the public workforce to label the data. Use the SageMaker Object Detection algorithm to train a model. Use SageMaker batch transform for inference.

Explanation:

Here's a detailed justification for why option A is the best solution, considering the company's limited ML expertise and the need for efficiency:

Option A is the most suitable because it leverages services that abstract away much of the ML complexity while still providing good accuracy. Amazon Rekognition Custom Labels is designed for users with limited ML experience. It simplifies the process of training a computer vision model with your own images. This is important because the internal team has no ML experience.

Using a private workforce composed of the internal team makes sense because they likely have domain knowledge about what solar panels look like in satellite imagery, which is essential for accurate labeling. While using a public workforce (option D) might seem faster, the lack of domain expertise could lead to poor labeling quality and ultimately a less effective model.

SageMaker Ground Truth's active learning feature further reduces effort. Active learning intelligently selects the most informative images for labeling, reducing the total number of images that need to be labeled to achieve a target accuracy. This optimizes the labeling process significantly.

SageMaker Object Detection, suggested in options C and D, requires more ML expertise to configure, train, and optimize than Rekognition Custom Labels. The team lacks this expertise, making these options less suitable. Batch transform in C and D is not needed as Rekognition Custom Labels hosting abstracts all of this.

Therefore, option A combines the accessibility of Rekognition Custom Labels with the efficiency of SageMaker Ground Truth's active learning and internal domain expertise, making it the best choice for a company with limited ML capabilities.

Here are some authoritative links for further research:

Amazon Rekognition Custom Labels: https://aws.amazon.com/rekognition/custom-labels-features/
Amazon SageMaker Ground Truth: https://aws.amazon.com/sagemaker/groundtruth/

Question: 265

Exam Heist

A company hosts a machine learning (ML) dataset repository on Amazon S3. A data scientist is preparing the repository to train a model. The data scientist needs to redact personally identifiable information (PH) from the dataset.

Which solution will meet these requirements with the LEAST development effort?

- Use Amazon SageMaker Data Wrangler with a custom transformation to identify and redact the PII.
- Create a custom AWS Lambda function to read the files, identify the PII. and redact the PII
- Use AWS Glue DataBrew to identity and redact the PII ✓
- Use an AWS Glue development endpoint to implement the PII redaction from within a notebook

Explanation:

Here's a detailed justification for why option C, using AWS Glue DataBrew, is the most suitable solution for redacting PII from an S3-based ML dataset with the least development effort:

AWS Glue DataBrew is a visual data preparation tool specifically designed to cleanse and normalize data, including identifying and redacting PII. It offers pre-built transformations and recipes to accomplish this task without writing custom code. DataBrew natively integrates with S3, allowing direct access to the ML dataset.

Option A, SageMaker Data Wrangler, can also perform data transformation, but requires configuring a custom transformation for PII redaction, involving more development effort compared to DataBrew's built-in functionalities.
Option B, a custom Lambda function, necessitates writing code to read files, identify PII using NLP or regular expressions, and then redact it. This demands significantly more development time and expertise than leveraging a managed service like DataBrew. Furthermore, a custom Lambda function would necessitate careful error handling, scaling considerations, and security management, increasing the operational overhead.
Option D, using an AWS Glue development endpoint, involves writing custom Spark code to achieve PII redaction. This is also more complex and time-consuming than DataBrew's pre-built capabilities.
DataBrew simplifies the PII redaction process through its visual interface and pre-built recipes. The data scientist can easily identify sensitive fields using DataBrew's data profiling capabilities and then apply redaction transformations like masking, suppression, or tokenization. These transformations are applied through a simple UI, avoiding the need for custom code and ensuring consistent PII handling across the entire dataset. The solution offers the fastest path to PII redaction with the least development effort, fitting the requirements.
Authoritative Links:

AWS Glue DataBrew: https://aws.amazon.com/glue/databrew/
AWS Glue DataBrew documentation: https://docs.aws.amazon.com/databrew/latest/dg/what-is.html

Question: 266

Exam Heist

A company is deploying a new machine learning (ML) model in a production environment. The company is concerned that the ML model will drift over time, so the company creates a script to aggregate all inputs and predictions into a single file at the end of each day. The company stores the file as an object in an Amazon S3 bucket. The total size of the daily file is 100 GB. The daily file size will increase over time.

Four times a year, the company samples the data from the previous 90 days to check the ML model for drift. After the 90-day period, the company must keep the files for compliance reasons.

The company needs to use S3 storage classes to minimize costs. The company wants to maintain the same storage durability of the data.

Which solution will meet these requirements?

- Store the daily objects in the S3 Standard-InfrequentAccess (S3 Standard-IA) storage class. Configure an S3 Lifecycle rule to move the objects to S3 Glacier Flexible Retrieval after 90 days.
- Store the daily objects in the S3 One Zone-Infrequent Access (S3 One Zone-IA) storage class. Configure an S3 Lifecycle rule to move the objects to S3 Glacier Flexible Retrieval after 90 days.
- Store the daily objects in the S3 Standard-InfrequentAccess (S3 Standard-IA) storage class. Configure an S3 Lifecycle rule to move the objects to S3 Glacier Deep Archive after 90 days. ✓
- Store the daily objects in the S3 One Zone-Infrequent Access (S3 One Zone-IA) storage class. Configure an S3 Lifecycle rule to move the objects to S3 Glacier Deep Archive after 90 days.

Explanation:
The correct answer is C. Here's why:
The problem requires a cost-effective storage solution for daily ML model input/prediction files that are initially accessed quarterly for drift analysis (90-day window) and then archived for long-term compliance.
Option A uses S3 Standard-IA initially, which is a good choice because the data is accessed infrequently (4 times a year). It then moves the data to S3 Glacier Flexible Retrieval after 90 days. While Glacier Flexible Retrieval is cheaper than S3 Standard-IA, Glacier Deep Archive is significantly cheaper and perfectly suited for infrequent access needed only for compliance after the 90 day period.
Option B uses S3 One Zone-IA initially. This storage class stores data in a single Availability Zone, which reduces costs but also significantly reduces durability. The requirement is to maintain the same storage durability, making this option unsuitable. S3 Standard, Standard-IA, Intelligent-Tiering, and Glacier storage classes all provide 99.999999999% (11 9's) of data durability because they store data across multiple Availability Zones.
Option D, similar to Option B, uses S3 One Zone-IA initially, failing to meet the durability requirement. It transitions data to Glacier Deep Archive after 90 days, which is cost-effective, but the initial storage class violates the stated durability constraint.
Option C starts with S3 Standard-IA for infrequent access during the first 90 days. After 90 days, the data moves to S3 Glacier Deep Archive, which is designed for long-term data archival and the lowest cost storage option, and therefore meets the compliance requirements. S3 Standard-IA provides high durability for the data for the first 90 days while it might need to be accessed for ML model drift analysis. Then, the data is transitioned to the lower cost, longer-term storage of S3 Glacier Deep Archive, which is ideal for compliance needs after the drift analysis window has closed.
Therefore, Option C provides the required durability with cost optimization.
Relevant links:

S3 Storage Classes: https://aws.amazon.com/s3/storage-classes/
S3 Lifecycle: https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html
Glacier Deep Archive: https://aws.amazon.com/glacier/deep-archive/

Question: 267

Exam Heist

A company wants to enhance audits for its machine learning (ML) systems. The auditing system must be able to perform metadata analysis on the features that the ML models use. The audit solution must generate a report that analyzes the metadata. The solution also must be able to set the data sensitivity and authorship of features.

Which solution will meet these requirements with the LEAST development effort?

- Use Amazon SageMaker Feature Store to select the features. Create a data flow to perform feature-level metadata analysis. Create an Amazon DynamoDB table to store feature-level metadata. Use Amazon QuickSight to analyze the metadata.
- Use Amazon SageMaker Feature Store to set feature groups for the current features that the ML models use. Assign the required metadata for each feature. Use SageMaker Studio to analyze the metadata.
- Use Amazon SageMaker Features Store to apply custom algorithms to analyze the feature-level metadata that the company requires. Create an Amazon DynamoDB table to store feature-level metadata. Use Amazon QuickSight to analyze the metadata.
- Use Amazon SageMaker Feature Store to set feature groups for the current features that the ML models use. Assign the required metadata for each feature. Use Amazon QuickSight to analyze the metadata. ✓

Explanation:
The correct answer is **D**. Here's why:
A. Incorrect. While this approach uses SageMaker Feature Store and QuickSight, it requires creating a separate data flow and DynamoDB table, adding unnecessary complexity and development effort. SageMaker Feature Store already provides the necessary metadata storage and QuickSight can directly analyze it.
B. Incorrect. SageMaker Studio is primarily an IDE for developing and managing ML models. While useful for some analysis, it's not designed for comprehensive, report-driven metadata analysis across features like QuickSight is. Furthermore, option D leverages QuickSight for metadata analysis, which is more suitable than SageMaker Studio.
C. Incorrect. This is also incorrect because it requires custom algorithms and an additional DynamoDB table. SageMaker Feature Store can natively handle much of the required metadata and QuickSight provides built-in analytics capabilities. It is unnecessary to implement custom algorithms and create a separate DynamoDB table.
D. Correct. SageMaker Feature Store is designed to store and manage features, including metadata like data sensitivity and authorship. By creating feature groups and assigning metadata to each feature, you directly leverage its capabilities. QuickSight can then directly connect to SageMaker Feature Store and analyze this metadata, creating the required audit reports with minimal development effort. This approach aligns with the principle of using managed services to reduce operational overhead. It provides a streamlined solution that minimizes development time and complexity. Feature Store offers managed storage and retrieval of feature data, making it well-suited for this task, and the metadata is directly accessible.
Key Concepts:

Amazon SageMaker Feature Store: A fully managed repository for ML features. It helps teams store, update, retrieve, and share features for training and inference.
Amazon QuickSight: A business intelligence service that allows you to create interactive dashboards and visualizations. It can connect to various data sources, including SageMaker Feature Store.
Supporting Links:

Amazon SageMaker Feature Store Documentation
Amazon QuickSight Documentation

Question: 268

Exam Heist

A machine learning (ML) specialist uploads a dataset to an Amazon S3 bucket that is protected by server-side encryption with AWS KMS keys (SSE-KMS). The ML specialist needs to ensure that an Amazon SageMaker notebook instance can read the dataset that is in Amazon S3.

Which solution will meet these requirements?

- Define security groups to allow all HTTP inbound and outbound traffic. Assign the security groups to the SageMaker notebook instance.
- Configure the SageMaker notebook instance to have access to the VPC. Grant permission in the AWS Key Management Service (AWS KMS) key policy to the notebook's VPC.

- Assign an IAM role that provides S3 read access for the dataset to the SageMaker notebook. Grant permission in the KMS key policy to the IAM role. ✓
- Assign the same KMS key that encrypts the data in Amazon S3 to the SageMaker notebook instance.

Explanation:

The correct answer is **C**. Here's a detailed justification:

Why C is Correct:

IAM Roles and Permissions: AWS Identity and Access Management (IAM) roles are the standard and secure way to grant permissions to AWS resources, like a SageMaker notebook instance. By assigning an IAM role to the notebook, you are providing it with temporary security credentials to access AWS services.

S3 Read Access: The IAM role should include a policy that grants read-only access (`s3:GetObject`) to the specific S3 bucket and objects (or a prefix within the bucket) containing the dataset.

KMS Key Policy: When using SSE-KMS encryption, the KMS key used to encrypt the data in S3 has a key policy that controls who can use the key for decryption. You must explicitly grant permission to the IAM role associated with the SageMaker notebook to use the KMS key for decrypting the S3 objects. This is done by adding a statement to the KMS key policy allowing the IAM role's ARN to perform `kms:Decrypt` operations.

Principle of Least Privilege: This approach follows the principle of least privilege, granting the notebook instance only the necessary permissions to access the specific dataset.

Why Other Options are Incorrect:

A: Opening all HTTP inbound and outbound traffic via security groups is a massive security risk. Security groups control network traffic, not access to encrypted data. This option is overly permissive and doesn't address the KMS encryption.

B: While connecting the SageMaker notebook to a VPC is often a good practice, it doesn't directly address the KMS encryption. Granting permission to the notebook's VPC in the KMS key policy is too broad. The more fine-grained approach is to give the notebook's IAM role permission. Also VPC in itself, does not have inherent access to a key, it is the services or instance in the VPC that get access via IAM Role.

D: You cannot directly "assign" a KMS key to a SageMaker notebook instance in the way the option suggests. KMS keys are not attached to instances like that. Furthermore, even if you could, it would be bad security practice. The correct approach is to manage access via IAM roles and the KMS key policy.

In summary, option C is the correct solution because it uses IAM roles for secure permission management and explicitly grants the notebook instance (through its role) the necessary permissions to both read the S3 data and decrypt it using the KMS key, following the principle of least privilege.

Authoritative Links:

IAM Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
SSE-KMS: https://docs.aws.amazon.com/AmazonS3/latest/userguide/kms-keyspec.html
KMS Key Policies: https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html
SageMaker IAM Roles: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html

Question: 269

Exam Heist

A company has a podcast platform that has thousands of users. The company implemented an algorithm to detect low podcast engagement based on a 10-minute running window of user events such as listening to, pausing, and closing the podcast. A machine learning (ML) specialist is designing the ingestion process for these events. The ML specialist needs to transform the data to prepare the data for inference.

How should the ML specialist design the transformation step to meet these requirements with the LEAST operational effort?

- Use an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to ingest event data. Use Amazon Kinesis Data Analytics to transform the most recent 10 minutes of data before inference.
- Use Amazon Kinesis Data Streams to ingest event data. Store the data in Amazon S3 by using Amazon Kinesis Data Firehose. Use AWS Lambda to transform the most recent 10 minutes of data before inference.
- Use Amazon Kinesis Data Streams to ingest event data. Use Amazon Kinesis Data Analytics to transform the most recent 10 minutes of data before inference. ✓
- Use an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster to ingest event data. Use AWS Lambda to transform the most recent 10 minutes of data before inference.

Explanation:

The correct answer is **C**. Here's a detailed justification:

Why C is the best choice:

Kinesis Data Streams for Ingestion: Kinesis Data Streams is a highly scalable and durable real-time data streaming service. It's perfectly suited for ingesting a continuous stream of user event data from the podcast platform. It is designed for high-velocity data ingestion.
Kinesis Data Analytics for Transformation: Kinesis Data Analytics allows you to process and analyze streaming data in real-time using SQL or Apache Flink. Its key advantage is its ability to perform windowed operations. A 10-minute running window is easily implemented within Kinesis Data Analytics using SQL or Flink constructs. This makes it ideal for transforming data within a defined timeframe.
Least Operational Effort: Kinesis Data Analytics provides a fully managed environment. This minimizes the operational overhead of managing servers, scaling infrastructure, or dealing with complex configuration. Its tight integration with Kinesis Data Streams further simplifies the overall architecture.
Why other options are less suitable:

A and D (MSK and Lambda/Kinesis Analytics): While MSK can be used for streaming data, it is more complex to manage compared to Kinesis Data Streams. You'd need to handle Kafka cluster setup, scaling, and maintenance. MSK is beneficial when you already have a Kafka ecosystem or require fine-grained control.
B (Kinesis Data Streams, S3, Firehose, Lambda): This solution introduces unnecessary complexity. Storing data in S3 using Firehose adds an extra step, and then using Lambda to transform the most recent 10 minutes of data is inefficient compared to Kinesis Data Analytics. Lambda is better suited for event-driven computations, not for continuous windowed transformations. Using S3 as an intermediate stage increases latency.
In summary: Kinesis Data Streams for ingestion and Kinesis Data Analytics for real-time windowed transformations provides the most straightforward, scalable, and operationally efficient solution for the given requirements. It minimizes operational overhead by leveraging managed services and simplifies data processing within the desired timeframe.
Authoritative Links:

Amazon Kinesis Data Streams: https://aws.amazon.com/kinesis/data-streams/
Amazon Kinesis Data Analytics: https://aws.amazon.com/kinesis/data-analytics/
Amazon Managed Streaming for Apache Kafka (Amazon MSK): https://aws.amazon.com/msk/
AWS Lambda: https://aws.amazon.com/lambda/
Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/

Question: 270

Exam Heist

A machine learning (ML) specialist is training a multilayer perceptron (MLP) on a dataset with multiple classes. The target class of interest is unique compared to the other classes in the dataset, but it does not achieve an acceptable recall metric. The ML specialist varies the number and size of the MLP's hidden layers, but the results do not improve significantly.

Which solution will improve recall in the LEAST amount of time?

- Add class weights to the MLP's loss function, and then retrain. ✓
- Gather more data by using Amazon Mechanical Turk, and then retrain.
- Train a k-means algorithm instead of an MLP.
- Train an anomaly detection model instead of an MLP.

Explanation:
The question highlights a scenario where an MLP model struggles with recall for a specific, unique class despite architecture tuning. The goal is to improve recall efficiently.
Option A, adding class weights to the MLP's loss function, is the most efficient approach. The problem likely stems from class imbalance – the target class being rare compared to others. Class weights penalize the model more heavily for misclassifying instances of the underrepresented class, forcing it to prioritize correctly identifying them and thus improving recall. This requires a simple code modification and retraining, which is relatively quick.
Option B, gathering more data with Amazon Mechanical Turk, is time-consuming and may not guarantee improved recall if the new data doesn't adequately represent the underrepresented class. Data acquisition, cleaning, and labeling are lengthy processes.
Option C, switching to k-means, is unsuitable. K-means is an unsupervised clustering algorithm designed for grouping data points based on similarity, not classification based on labeled data, rendering it inappropriate for the stated problem. K-means operates without knowledge of the target class, making it ineffective for improving recall for a specific label.
Option D, training an anomaly detection model, might be a viable alternative if the target class represents anomalous data. However, it's less direct than addressing the class imbalance. Furthermore, anomaly detection typically identifies outliers *in general* rather than focusing specifically on improving recall for the *target* class, so it might require significant tweaking. Anomaly detection techniques may not directly maximize the recall of a particular class label like class weighting can.

Therefore, adjusting class weights in the loss function addresses the imbalanced class problem directly with minimal effort, making it the best solution in terms of time and effectiveness. It allows the model to learn more effectively from the existing dataset by emphasizing the importance of the underrepresented class during training.
https://developers.google.com/machine-learning/crash-course/classification/imbalanced-datahttps://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

Question: 271

Exam Heist

A machine learning (ML) specialist uploads 5 TB of data to an Amazon SageMaker Studio environment. The ML specialist performs initial data cleansing. Before the ML specialist begins to train a model, the ML specialist needs to create and view an analysis report that details potential bias in the uploaded data.

Which combination of actions will meet these requirements with the LEAST operational overhead? (Choose two.)

- [x] Use SageMaker Clarify to automatically detect data bias ✓
- [ ] Turn on the bias detection option in SageMaker Ground Truth to automatically analyze data features.
- [ ] Use SageMaker Model Monitor to generate a bias drift report.
- [x] Configure SageMaker Data Wrangler to generate a bias report. ✓
- [ ] Use SageMaker Experiments to perform a data check

Explanation:
The requirement is to analyze data for potential bias *before* training a model with minimal operational overhead.
A. Use SageMaker Clarify to automatically detect data bias: This is a correct choice. SageMaker Clarify is specifically designed to detect potential bias in datasets and models. It can be run directly on the raw dataset *before* training, fulfilling the requirement. It provides detailed reports on various bias metrics. https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-detect-data-bias.html
D. Configure SageMaker Data Wrangler to generate a bias report: This is also a suitable choice. SageMaker Data Wrangler provides a visual interface for data preparation and analysis. It includes built-in bias analysis capabilities, allowing users to generate reports that highlight potential biases in the dataset before model training. Data Wrangler operates within the SageMaker Studio environment making it easy to access the data and analyze for bias.
https://docs.aws.amazon.com/sagemaker/latest/dg/data-wrangler-bias.html
Why other options are incorrect:

B. Turn on the bias detection option in SageMaker Ground Truth to automatically analyze data features: SageMaker Ground Truth is used for data labeling, not for bias detection in the raw dataset.
C. Use SageMaker Model Monitor to generate a bias drift report: SageMaker Model Monitor is used to detect data and model drift after the model has been deployed and is making predictions. It doesn't analyze the initial dataset for bias *before* training.
E. Use SageMaker Experiments to perform a data check: SageMaker Experiments is primarily for tracking and managing machine learning experiments. While it can track various metrics, it doesn't provide specific built-in features for bias detection in the initial dataset.
Therefore, using SageMaker Clarify or SageMaker Data Wrangler offers the most direct and least operationally heavy approach to analyzing the data for bias prior to model training. Both tools are designed to provide detailed reports and operate within or alongside the SageMaker Studio environment, minimizing the effort required to set up and run the analysis.

Question: 272

Exam Heist

A network security vendor needs to ingest telemetry data from thousands of endpoints that run all over the world. The data is transmitted every 30 seconds in the form of records that contain 50 fields. Each record is up to 1 KB in size. The security vendor uses Amazon Kinesis Data Streams to ingest the data. The vendor requires hourly summaries of the records that Kinesis Data Streams ingests. The vendor will use Amazon Athena to query the records and to generate the summaries. The Athena queries will target 7 to 12 of the available data fields.

Which solution will meet these requirements with the LEAST amount of customization to transform and store the ingested data?

- ( ) Use AWS Lambda to read and aggregate the data hourly. Transform the data and store it in Amazon S3 by using Amazon Kinesis Data Firehose.
- ( ) Use Amazon Kinesis Data Firehose to read and aggregate the data hourly. Transform the data and store it in Amazon S3 by using a short-lived Amazon EMR cluster.
- (x) Use Amazon Kinesis Data Analytics to read and aggregate the data hourly. Transform the data and store it in Amazon S3 by using Amazon Kinesis Data Firehose. ✓

- Use Amazon Kinesis Data Firehose to read and aggregate the data hourly. Transform the data and store it in Amazon S3 by using AWS Lambda.

Explanation:
The correct answer is C because it utilizes Kinesis Data Analytics for in-stream aggregation, minimizing the need for custom code and simplifying the data transformation and storage process. Here's a breakdown:

Kinesis Data Streams: Already in place for ingestion, so no changes needed there.
Hourly Summaries: Kinesis Data Analytics is specifically designed for real-time data processing and analytics. It can aggregate the data into hourly summaries using SQL-based queries directly from the stream. This fulfills the requirement of hourly summaries with minimal coding effort. https://aws.amazon.com/kinesis/data-analytics/
Transformation and Storage: Kinesis Data Firehose excels at reliably delivering streaming data to various destinations, including S3. Kinesis Data Analytics can transform the data (selecting the 7-12 fields needed) and then send the aggregated summaries to Kinesis Data Firehose. Firehose will then deliver it to S3. https://aws.amazon.com/kinesis/data-firehose/
Athena: Athena is used to query the records and generate summaries.
Here's why the other options are less optimal:

A: Lambda for Aggregation: While Lambda *could* aggregate, it would require significantly more custom code to manage the aggregation logic and potential scaling issues with high throughput. It is a better practice to use services designed for stream processing, like Kinesis Data Analytics. Furthermore, Lambda would have to connect directly to Kinesis Data Streams.
B: Firehose for Aggregation with EMR: Kinesis Data Firehose primarily focuses on delivering data and performing basic transformations (like format conversion). It's not designed for complex aggregations like hourly summaries. EMR is overkill for this scenario; it is expensive and more complex than Kinesis Data Analytics.
D: Firehose for Aggregation with Lambda: As with option B, Firehose is not optimal for aggregation. Using Lambda here requires manually managing aggregation logic and potential scaling issues. This is a less elegant and more complex solution than Kinesis Data Analytics. Using Lambda after firehose for transformation can result in increased latency and complexity. Therefore, Kinesis Data Analytics provides the most efficient and cost-effective solution, minimizing the need for custom code and simplifying the overall architecture.

Question: 273

Exam Heist

A machine learning (ML) specialist is training a linear regression model. The specialist notices that the model is overfitting. The specialist applies an L1 regularization parameter and runs the model again. This change results in all features having zero weights.

What should the ML specialist do to improve the model results?

- Increase the L1 regularization parameter. Do not change any other training parameters.
- Decrease the L1 regularization parameter. Do not change any other training parameters. ✓
- Introduce a large L2 regularization parameter. Do not change the current L1 regularization value.
- Introduce a small L2 regularization parameter. Do not change the current L1 regularization value.

Explanation:
The ML specialist's model overfits, and applying L1 regularization results in all features having zero weights. This means the L1 regularization penalty is too strong. L1 regularization, also known as Lasso regression, adds a penalty proportional to the absolute value of the coefficients to the loss function. This encourages sparsity in the model by driving the coefficients of less important features to zero.
When the L1 regularization parameter is too high, the penalty for having non-zero coefficients becomes excessive, leading the model to minimize the overall cost by setting all coefficients to zero. In essence, the model is prioritizing minimizing the penalty term over fitting the data.
Therefore, to improve the model, the specialist should decrease the L1 regularization parameter. Reducing the parameter lowers the penalty for having non-zero coefficients, allowing the model to find a balance between fitting the data and keeping the coefficients small to prevent overfitting. Options C and D, which introduce L2 regularization, could be used in conjunction with L1 (Elastic Net), but the *immediate* problem is that L1 is too high and needs to be addressed first. While L2 regularization (Ridge regression) also helps prevent overfitting, introducing it without adjusting the excessive L1 penalty is unlikely to solve the root issue. Option A would only exacerbate the problem by making the L1 penalty even stronger.
Therefore, the most appropriate action is to reduce the L1 parameter to allow for some feature coefficients to be non-zero and contribute to a more accurate model.
Further research:

L1 Regularization (Lasso): https://en.wikipedia.org/wiki/Lasso_(statistics)

L2 Regularization (Ridge): https://en.wikipedia.org/wiki/Tikhonov_regularization
Regularization in Machine Learning: https://www.geeksforgeeks.org/regularization-in-machine-learning/

Question: 274

Exam Heist

A data scientist for a medical diagnostic testing company has developed a machine learning (ML) model to identify patients who have a specific disease. The dataset that the scientist used to train the model is imbalanced. The dataset contains a large number of healthy patients and only a small number of patients who have the disease. The model should consider that patients who are incorrectly identified as positive for the disease will increase costs for the company.

Which metric will MOST accurately evaluate the performance of this model?

- Recall
- F1 score
- Accuracy
- Precision ✓

Explanation:

Here's a detailed justification for why Precision is the most accurate metric in this scenario, given the imbalanced dataset and the high cost associated with false positives:

The core issue is the imbalanced dataset, where there are significantly more healthy patients than patients with the disease. This skewness can lead to misleadingly high accuracy scores if the model simply predicts "healthy" for most patients. Accuracy alone doesn't tell us how well the model identifies the minority class (patients with the disease). In this specific case, the cost of a false positive (incorrectly identifying a healthy patient as having the disease) is high.

Precision focuses on the proportion of positive predictions that are actually correct. It answers the question: "Of all the patients the model predicted as having the disease, how many actually have the disease?". A high precision score means the model is making fewer false positive predictions. This is critical because false positives lead to unnecessary further testing, increasing costs. Recall, on the other hand, focuses on the proportion of actual positive cases that the model correctly identifies. It prioritizes minimizing false negatives, but this is less important given the prompt's emphasis on reducing the financial burden of false positives. The F1 score is the harmonic mean of precision and recall, balancing both. However, since the question highlights the disproportionate cost of false positives, weighting precision is more appropriate than a balanced metric.

Therefore, in a situation with imbalanced data and high costs associated with false positives, precision directly addresses the business concern by prioritizing the accuracy of positive predictions. It provides the most relevant insight into whether the model is effectively minimizing unnecessary expenditures associated with incorrectly classifying healthy patients. This makes precision the most suitable metric in evaluating the model's performance.

Authoritative Links:

Precision and Recall: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall
Imbalanced Data: https://www.analyticsvidhya.com/blog/2020/07/handling-imbalanced-dataset/

Question: 275

Exam Heist

A wildlife research company has a set of images of lions and cheetahs. The company created a dataset of the images. The company labeled each image with a binary label that indicates whether an image contains a lion or cheetah. The company wants to train a model to identify whether new images contain a lion or cheetah.

Which Amazon SageMaker algorithm will meet this requirement?

- XGBoost
- Image Classification - TensorFlow ✓
- Object Detection - TensorFlow
- Semantic segmentation - MXNet

Explanation:

The correct answer is **B. Image Classification - TensorFlow**. Here's a detailed justification:

The problem describes a binary classification task where the goal is to categorize images into one of two classes: lion or cheetah. This aligns perfectly with the purpose of image classification algorithms. Image classification algorithms learn to associate visual features in images with specific class labels. In this case, the algorithm will learn to distinguish between the visual features of lions and cheetahs.
Option A, XGBoost, is a powerful gradient boosting algorithm often used for tabular data, but it's not directly applicable to image data without feature extraction. While XGBoost *could* be used if image features were pre-extracted (e.g., using a Convolutional Neural Network or CNN), it's not the most efficient or direct approach for image classification.
Option C, Object Detection - TensorFlow, is designed to identify and locate multiple objects within an image using bounding boxes, along with classification. This is an overkill for the requirement, as the company only needs to determine if a single image contains a lion or cheetah, without locating them.
Option D, Semantic Segmentation - MXNet, involves classifying each pixel in an image, assigning it to a particular class. This is a much more granular task than simply classifying the entire image as containing a lion or cheetah, adding unnecessary complexity to the task.
Image Classification algorithms, particularly those implemented with TensorFlow in Amazon SageMaker, are specifically designed for these kinds of image classification tasks. TensorFlow provides a rich set of tools and pre-trained models that can be fine-tuned for this specific use case.
Therefore, **Image Classification - TensorFlow** provides the most efficient and direct solution for training a model to identify whether new images contain a lion or cheetah.
Authoritative Links:

Amazon SageMaker Image Classification: https://docs.aws.amazon.com/sagemaker/latest/dg/image-classification.html
TensorFlow: https://www.tensorflow.org/

Question: 276

Exam Heist

A company is planning a marketing campaign to promote a new product to existing customers. The company has data for past promotions that are similar. The company decides to try an experiment to send a more expensive marketing package to a smaller number of customers. The company wants to target the marketing campaign to customers who are most likely to buy the new product. The experiment requires that at least 90% of the customers who are likely to purchase the new product receive the marketing materials.

The company trains a model by using the linear learner algorithm in Amazon SageMaker. The model has a recall score of 80% and a precision of 75%.

How should the company retrain the model to meet these requirements?

- Set the target_recall hyperparameter to 90%. Set the binary_classifier_model_selection_criteria hyperparameter to recall_at_target_precision. ✓
- Set the target_precision hyperparameter to 90%. Set the binary_classifier_model_selection_criteria hyperparameter to precision_at_target_recall.
- Use 90% of the historical data for training. Set the number of epochs to 20.
- Set the normalize_label hyperparameter to true. Set the number of classes to 2.

Explanation:
The company needs to improve the model's recall to ensure that at least 90% of likely purchasers are targeted. Recall is the metric that measures the model's ability to find all the relevant cases (customers likely to buy). Option A directly addresses this by using the `target_recall` hyperparameter in SageMaker's Linear Learner. Setting `target_recall` to 90% instructs the algorithm to optimize the model to achieve at least this recall level. The `binary_classifier_model_selection_criteria` hyperparameter, when set to `recall_at_target_precision`, tells the algorithm to select the model that maximizes recall while trying to maintain a certain level of precision.
Option B is incorrect because it focuses on precision, which isn't the primary concern. The requirement is to reach at least 90% of the target customers, indicated by Recall. Option C changes the training dataset size and number of epochs, but it doesn't directly address the Recall performance. Option D adjusts data normalization and number of classes, and it doesn't have a direct impact on Recall.
Linear Learner's `target_recall` is specifically designed to directly influence the model's Recall. The `binary_classifier_model_selection_criteria` is a means to control the trade-off between Precision and Recall. If the company were to use default settings, it may find a model with 80% Recall is the "best". By explicitly setting a `target_recall`, the model is optimized for *at least* the specified value.
Refer to the AWS documentation for more details on the Linear Learner algorithm and its hyperparameters. The SageMaker documentation provides valuable insights into how to tune the model for desired metrics.
Amazon SageMaker Linear Learner AlgorithmSageMaker Hyperparameter Tuning

Question: 277

Exam Heist

A company's machine learning (ML) specialist is building a computer vision model to classify 10 different traffic signs. The company has stored 100 images of each class in Amazon S3, and the company has another 10,000 unlabeled images. All the images come from dash cameras and are a size of 224 pixels × 224 pixels. After several training runs, the model is overfitting on the training data.

Which actions should the ML specialist take to address this problem? (Choose two.)

- [x] Use Amazon SageMaker Ground Truth to label the unlabeled images. ✓
- [ ] Use image preprocessing to transform the images into grayscale images.
- [x] Use data augmentation to rotate and translate the labeled images. ✓
- [ ] Replace the activation of the last layer with a sigmoid.
- [ ] Use the Amazon SageMaker k-nearest neighbors (k-NN) algorithm to label the unlabeled images.

Explanation:

The problem is model overfitting, meaning the model is performing well on the training data but poorly on unseen data. This often arises when the model learns the training data too well, including its noise and specific characteristics, failing to generalize.

Option A, using Amazon SageMaker Ground Truth to label the unlabeled images, is a good choice. Adding more labeled data (in this case, converting some of the 10,000 unlabeled images into labeled data) can help the model generalize better. More data allows the model to learn a more robust representation of the underlying patterns, reducing the impact of noise or specific quirks in the initial training set. This addresses overfitting by providing the model with more examples to learn from. SageMaker Ground Truth simplifies the labeling process with features such as automated labeling and workforce management. [https://aws.amazon.com/sagemaker/groundtruth/]

Option C, using data augmentation to rotate and translate the labeled images, also combats overfitting. Data augmentation artificially increases the size of the training dataset by creating modified versions of existing images. Rotating, translating, scaling, or adding noise to the existing images helps the model become more robust to variations in the input data. By exposing the model to these different perspectives of the same underlying class, it learns to focus on the core features and become less sensitive to specific orientations, positions, or sizes. This, in turn, reduces overfitting and improves generalization.

Option B, transforming the images into grayscale, might reduce computational complexity but it also significantly reduces the information available to the model. Color can be a distinguishing feature in identifying traffic signs, so removing it could hinder the model's ability to learn effectively, potentially leading to underfitting rather than solving overfitting.

Option D, replacing the activation of the last layer with a sigmoid, is inappropriate. The model classifies 10 different traffic signs, which is a multi-class classification problem. Sigmoid activation is typically used for binary classification problems. Therefore, this change will not improve the model's generalization ability, and is likely to break the output altogether.

Option E, using Amazon SageMaker k-NN to label unlabeled images, presents a potentially problematic and less accurate approach. While k-NN can be used for classification, labeling images with a k-NN algorithm first requires building a k-NN model based on the existing, and potentially already biased, dataset. If the original data causes overfitting, the k-NN model will also likely reflect these biases, perpetuating the problem of overfitting when labeling the new images. Furthermore, it doesn't directly address the overfitting problem itself. Active learning through SageMaker Ground Truth, using a human-in-the-loop, will lead to better data quality.

Question: 278

Exam Heist

A data science team is working with a tabular dataset that the team stores in Amazon S3. The team wants to experiment with different feature transformations such as categorical feature encoding. Then the team wants to visualize the resulting distribution of the dataset. After the team finds an appropriate set of feature transformations, the team wants to automate the workflow for feature transformations.

Which solution will meet these requirements with the MOST operational efficiency?

- ◯ Use Amazon SageMaker Data Wrangler preconfigured transformations to explore feature transformations. Use SageMaker Data Wrangler templates for visualization. Export the feature processing workflow to a SageMaker pipeline for automation. ✓
- ◯ Use an Amazon SageMaker notebook instance to experiment with different feature transformations. Save the transformations to Amazon S3. Use Amazon QuickSight for visualization. Package the feature processing steps into an AWS Lambda function for automation.
- ◯ Use AWS Glue Studio with custom code to experiment with different feature transformations. Save the transformations to Amazon S3. Use Amazon QuickSight for visualization. Package the feature processing steps into an AWS Lambda function for automation.
- ◯ Use Amazon SageMaker Data Wrangler preconfigured transformations to experiment with different feature transformations. Save the transformations to Amazon S3. Use Amazon QuickSight for visualization. Package each feature transformation step into a separate AWS Lambda function. Use AWS Step Functions for workflow automation.

Explanation:
Option A is the most operationally efficient solution because it leverages the capabilities of Amazon SageMaker Data Wrangler for end-to-end feature engineering, visualization, and workflow automation.
SageMaker Data Wrangler provides preconfigured transformations that simplify feature engineering without requiring custom code for common tasks like categorical encoding. This reduces development effort and the risk of errors. Data Wrangler also includes built-in visualization tools for exploring data distributions, making it easier to assess the impact of different transformations directly within the tool.
The key advantage of Data Wrangler lies in its ability to export the feature processing workflow as a SageMaker Pipeline. This allows for seamless automation of the feature engineering process. A SageMaker Pipeline is a managed service that provides a reliable and scalable way to orchestrate machine learning workflows.
Option B involves using a SageMaker notebook instance and QuickSight for visualization. While this approach is feasible, it requires manually coding the feature transformations and managing the dependencies. Packaging feature processing steps into a Lambda function adds complexity, as Lambda functions have execution time limits and might not be suitable for large datasets or complex transformations.
Option C, using AWS Glue Studio with custom code, also involves manual coding and management. While Glue is excellent for ETL tasks, it may require significant effort to integrate with other services for visualization and automation. The process of deploying into Lambda faces similar limitations as described for option B.
Option D uses Data Wrangler but stores transformations manually in S3 and uses Lambda functions and Step Functions. While workable, it does not utilize Data Wrangler's full automation potential by exporting the workflow as a managed SageMaker pipeline. Lambda functions are also less efficient for this use case.
Therefore, the most efficient solution utilizes SageMaker Data Wrangler's capabilities for feature transformation, visualization, and workflow automation, eliminating the need for extensive custom coding, manual management, and reliance on Lambda for complex processing. This streamlines the entire process and minimizes operational overhead.
Supporting Links:

Amazon SageMaker Data Wrangler
Amazon SageMaker Pipelines

Question: 279

Exam Heist

A company plans to build a custom natural language processing (NLP) model to classify and prioritize user feedback. The company hosts the data and all machine learning (ML) infrastructure in the AWS Cloud. The ML team works from the company's office, which has an IPsec VPN connection to one VPC in the AWS Cloud.

The company has set both the enableDnsHostnames attribute and the enableDnsSupport attribute of the VPC to true. The company's DNS resolvers point to the VPC DNS. The company does not allow the ML team to access Amazon SageMaker notebooks through connections that use the public internet. The connection must stay within a private network and within the AWS internal network.

Which solution will meet these requirements with the LEAST development effort?

- Create a VPC interface endpoint for the SageMaker notebook in the VPC. Access the notebook through a VPN connection and the VPC endpoint. ✓
- Create a bastion host by using Amazon EC2 in a public subnet within the VPC. Log in to the bastion host through a VPN connection. Access the SageMaker notebook from the bastion host.
- Create a bastion host by using Amazon EC2 in a private subnet within the VPC with a NAT gateway. Log in to the bastion host through a VPN connection. Access the SageMaker notebook from the bastion host.
- Create a NAT gateway in the VPC. Access the SageMaker notebook HTTPS endpoint through a VPN connection and the NAT gateway.

Explanation:
The correct answer is **A. Create a VPC interface endpoint for the SageMaker notebook in the VPC. Access the notebook through a VPN connection and the VPC endpoint.**
Here's why:

VPC Interface Endpoints for SageMaker: VPC interface endpoints provide private connectivity to AWS services, including SageMaker, without traversing the public internet. This aligns directly with the requirement to avoid public internet access for SageMaker notebooks. They use AWS PrivateLink to establish a private connection between your VPC and the SageMaker service.

https://docs.aws.amazon.com/sagemaker/latest/dg/interface-vpc-endpoint.html
https://docs.aws.amazon.com/vpc/latest/privatelink/vpc-endpoints.html

VPN Connectivity: The company already has a VPN connection to the VPC. This established private connection is leveraged to securely access the VPC interface endpoint for SageMaker.
Least Development Effort: Option A requires creating a VPC endpoint, configuring security groups, and ensuring DNS resolution works correctly (which is already handled by the existing DNS setup). This is significantly less complex than setting up and managing bastion hosts or NAT gateways.
Let's analyze why the other options are not optimal:

B. Bastion Host in a Public Subnet: This option defeats the purpose of avoiding public internet access. While the connection to the bastion host is via VPN, the bastion host itself needs public internet access to reach SageMaker, which contradicts the requirement.
C. Bastion Host in a Private Subnet with NAT Gateway: This is an overly complex setup. While it avoids direct public internet exposure for the bastion host, a NAT gateway is primarily for outbound internet access from private subnets. It does not provide a private connection to SageMaker in the same way as a VPC endpoint, and requires more configuration.
D. NAT Gateway and SageMaker HTTPS Endpoint: This option exposes SageMaker through its HTTPS endpoint, which, while encrypted, is still accessible over the public internet. This violates the explicit requirement to keep the connection within the private AWS network. Also, it leverages the public endpoint for Sagemaker which is not needed since we have VPC already.
Therefore, creating a VPC interface endpoint is the most efficient and secure way to fulfill the requirement of private network access to SageMaker notebooks for the ML team. It minimizes development effort by leveraging the existing VPN connection and utilizes AWS PrivateLink for a direct, private connection to the SageMaker service.

Question: 280

Exam Heist

A data scientist is using Amazon Comprehend to perform sentiment analysis on a dataset of one million social media posts.

Which approach will process the dataset in the LEAST time?

- Use a combination of AWS Step Functions and an AWS Lambda function to call the DetectSentiment API operation for each post synchronously.
- Use a combination of AWS Step Functions and an AWS Lambda function to call the BatchDetectSentiment API operation with batches of up to 25 posts at a time. ✓
- Upload the posts to Amazon S3. Pass the S3 storage path to an AWS Lambda function that calls the StartSentimentDetectionJob API operation.
- Use an AWS Lambda function to call the BatchDetectSentiment API operation with the whole dataset.

Explanation:
The correct answer is B because it leverages the batch processing capabilities of Amazon Comprehend, offering a significant performance advantage over single-record processing and asynchronous batch jobs for the described scenario. Let's break down why the other options are less optimal:

Option A: Calling the `DetectSentiment` API for *each* post synchronously is incredibly inefficient. It incurs substantial overhead due to the individual API calls and will be significantly slower than batching. Each Lambda invocation has its own initialization and execution time, multiplying the overall processing duration by one million.
Option C: While `StartSentimentDetectionJob` is designed for large datasets, it's an *asynchronous* operation. The Lambda function would initiate the job and then terminate, without waiting for the job to complete. The data scientist would then have to poll for completion, adding complexity and potentially not being the *least* time. This method is suitable for extremely large datasets where immediate results are not required, which isn't necessarily implied in the prompt. S3 also needs to be properly configured and the data needs to be in a specific format.
Option D: While using `BatchDetectSentiment` is better than single record processing, a single Lambda function likely faces memory and execution time limitations when trying to process the *entire* one million post dataset. Lambda functions have execution time limits (default 15 minutes, can be increased), and memory limitations, meaning it will almost certainly fail. Additionally, it is unlikely that the entire one million posts can even be sent as an API request due to size limitations.
Option B combines the efficiency of batch processing (up to 25 posts per batch with `BatchDetectSentiment`) with the orchestration capabilities of AWS Step Functions. Step Functions can manage the iterative calls to the Lambda function, processing the dataset in chunks until all posts are analyzed. This balances the advantages of batch processing without overwhelming individual Lambda function executions or facing the Lambda limitations. This gives a much faster throughput than single calls but also avoids issues related to Lambda time and size limits.
In Summary:
Option B is the fastest and most appropriate for the scenario because it utilizes the efficient `BatchDetectSentiment` API within the Lambda function, orchestrating the batches through AWS Step Functions to avoid the limits of single Lambda functions, and it has the advantage of synchronous processing without the drawbacks of excessive individual API calls or the

asynchronous complexity of `StartSentimentDetectionJob`.The other options are less efficient, potentially exceeding memory limits or causing timeouts.

Supporting Links:

Amazon Comprehend Batch Processing: https://docs.aws.amazon.com/comprehend/latest/dg/batching-processing.html
AWS Step Functions: https://aws.amazon.com/step-functions/
AWS Lambda Limits: https://docs.aws.amazon.com/lambda/latest/dg/configuration-function-common.html

Question: 281

Exam Heist

A machine learning (ML) specialist at a retail company must build a system to forecast the daily sales for one of the company's stores. The company provided the ML specialist with sales data for this store from the past 10 years. The historical dataset includes the total amount of sales on each day for the store. Approximately 10% of the days in the historical dataset are missing sales data.

The ML specialist builds a forecasting model based on the historical dataset. The specialist discovers that the model does not meet the performance standards that the company requires.

Which action will MOST likely improve the performance for the forecasting model?

- Aggregate sales from stores in the same geographic area.
- Apply smoothing to correct for seasonal variation. ✓
- Change the forecast frequency from daily to weekly.
- Replace missing values in the dataset by using linear interpolation.

Explanation:

The correct answer is **B. Apply smoothing to correct for seasonal variation.**
Here's a detailed justification:
The core problem is that the forecasting model's performance is below required standards when predicting daily sales. Time series data, like daily sales, often exhibits seasonality – patterns that repeat over fixed periods (e.g., higher sales on weekends or during holidays). If the model isn't accounting for these repeating patterns, its performance will suffer.
Applying smoothing techniques (like moving averages, exponential smoothing, or Seasonal ARIMA) directly addresses this issue. Smoothing identifies and reduces noise/random fluctuations within the data, highlighting underlying trends and seasonal components. This makes the seasonal patterns more discernible to the model, improving forecast accuracy.
Smoothing essentially reduces the impact of outliers or irregular values within each season, emphasizing the typical seasonal behavior.
Option A (Aggregate sales from stores in the same geographic area) might be helpful if the data from a single store is too sparse or noisy. However, it's less directly related to addressing the fundamental problem of seasonality. Also, this might introduce biases or irrelevant data if stores have different demographics or operate under different local conditions.
Option C (Change the forecast frequency from daily to weekly) reduces the granularity of the forecast. This might improve performance metrics simply because it's easier to predict weekly aggregates than daily values (effectively hiding some of the variability). However, the business requirement is to predict *daily* sales. Changing the frequency is not solving the underlying issue of model performance; instead, it's side-stepping it.
Option D (Replace missing values in the dataset by using linear interpolation) is a useful data preprocessing step for handling missing data, especially if the missing data represents some continuous trend. Although missing data is definitely something to handle, it addresses a different issue than seasonal variation. Linear interpolation addresses the *completeness* of the data, and applying it would only provide a linear guess for missing points. While it fills in gaps, it wouldn't capture the complex seasonal variations within the data. Given that only 10% of the data is missing, the impact of improving missing value imputation is unlikely to be as significant as dealing with seasonal variation.
Therefore, applying smoothing is the MOST direct and likely method to enhance the performance of the forecasting model given the presence of seasonality.
Authoritative links:

Time Series Forecasting: https://otexts.com/fpp3/
Exponential Smoothing: https://en.wikipedia.org/wiki/Exponential_smoothing
ARIMA: https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

Question: 282

Exam Heist

A mining company wants to use machine learning (ML) models to identify mineral images in real time. A data science team built an image recognition model that is based on convolutional neural network (CNN). The team trained the model on Amazon SageMaker by using GPU instances.

The team will deploy the model to a SageMaker endpoint.

The data science team already knows the workload traffic patterns. The team must determine instance type and configuration for the workloads.

Which solution will meet these requirements with the LEAST development effort?

- Register the model artifact and container to the SageMaker Model Registry. Use the SageMaker Inference Recommender Default job type. Provide the known traffic pattern for load testing to select the best instance type and configuration based on the workloads.
- Register the model artifact and container to the SageMaker Model Registry. Use the SageMaker Inference Recommender Advanced job type. Provide the known traffic pattern for load testing to select the best instance type and configuration based on the workloads. ✓
- Deploy the model to an endpoint by using GPU instances. Use AWS Lambda and Amazon API Gateway to handle invocations from the web. Use open-source tools to perform load testing against the endpoint and to select the best instance type and configuration.
- Deploy the model to an endpoint by using CPU instances. Use AWS Lambda and Amazon API Gateway to handle invocations from the web. Use open-source tools to perform load testing against the endpoint and to select the best instance type and configuration.

Explanation:

The correct answer is **B**. Here's a detailed justification:

SageMaker Inference Recommender is designed to automate the process of finding the optimal instance type and configuration for deploying ML models. It eliminates the need for manual load testing and trial-and-error deployments. The problem states that the data science team wants the solution with the *least development effort*.

Option B leverages the SageMaker Inference Recommender with the *Advanced* job type. This type is more suitable when you have specific traffic patterns and performance requirements, as it runs load tests against your model to simulate real-world conditions and recommends the most cost-effective and performant instance configurations. Using known traffic patterns improves the accuracy of recommendations.

Option A's *Default* job type is a quicker option but doesn't allow for detailed configuration based on the specific workload and traffic patterns. This may lead to an under- or over-provisioned endpoint.

Options C and D introduce unnecessary complexity. They suggest deploying the model to a standard SageMaker endpoint and using AWS Lambda and API Gateway to handle invocations. While Lambda and API Gateway can be used to expose ML models, using them solely for inference testing and selection adds significant overhead and increases development effort. Additionally, using open-source load testing tools would require manual configuration, execution, and analysis of the results, making the process time-consuming and increasing the chances of error. Furthermore, option D is incorrect as the model was trained on GPU instances, using CPU instances for inference would likely lead to poor performance.

In summary, using the SageMaker Inference Recommender's *Advanced* job type allows the data science team to automate the process of finding the best instance type and configuration for their model, given their known traffic patterns. This approach ensures optimal performance and cost efficiency with minimal manual effort, satisfying all the requirements.

Authoritative Links:

SageMaker Inference Recommender: https://docs.aws.amazon.com/sagemaker/latest/dg/inference-recommender.html
Inference Recommender Job Types: https://docs.aws.amazon.com/sagemaker/latest/dg/inference-recommender-job-types.html

Question: 283

Exam Heist

A company is building custom deep learning models in Amazon SageMaker by using training and inference containers that run on Amazon EC2 instances. The company wants to reduce training costs but does not want to change the current architecture. The SageMaker training job can finish after interruptions. The company can wait days for the results.

Which combination of resources should the company use to meet these requirements MOST cost-effectively? (Choose two.)

- [ ] On-Demand Instances
- [x] Checkpoints ✓
- [ ] Reserved Instances
- [ ] Incremental training
- [x] Spot instances ✓

Explanation:

The optimal strategy for minimizing SageMaker training costs while tolerating interruptions and long completion times involves leveraging Spot Instances combined with Checkpointing.

E. Spot Instances: Spot Instances offer significantly lower pricing compared to On-Demand Instances because they utilize spare EC2 capacity. This allows the company to drastically reduce training costs. However, Spot Instances can be interrupted when AWS needs the capacity back. Given the company's tolerance for interruptions and ability to wait days for results, this

risk is acceptable.

Amazon EC2 Spot Instances

B. Checkpoints: Checkpointing is crucial for mitigating the impact of Spot Instance interruptions. By periodically saving the model's training state (i.e., weights, optimizer state), the training job can be resumed from the last checkpoint instead of starting from scratch after an interruption. This avoids wasting progress and reduces the overall training time and cost.

SageMaker Checkpointing

Why other options are not optimal:

A. On-Demand Instances: These are the most expensive option and don't contribute to cost reduction.
C. Reserved Instances: Reserved Instances require a commitment for a specific instance type and duration. While they offer savings compared to On-Demand, they don't address the need to tolerate interruptions or optimize for unused EC2 capacity like Spot Instances.
D. Incremental training: While incremental training can reduce costs in some situations by building on previous models, it is not as effective as Spot Instances and checkpointing for mitigating interruptions and optimizing for low cost and the approach requires a suitable base model, which might not always be available or applicable and checkpointing offers more direct benefit in the scenario of interruptible training.

Question: 284

Exam Heist

A company hosts a public web application on AWS. The application provides a user feedback feature that consists of free-text fields where users can submit text to provide feedback. The company receives a large amount of free-text user feedback from the online web application. The product managers at the company classify the feedback into a set of fixed categories including user interface issues, performance issues, new feature request, and chat issues for further actions by the company's engineering teams.

A machine learning (ML) engineer at the company must automate the classification of new user feedback into these fixed categories by using Amazon SageMaker. A large set of accurate data is available from the historical user feedback that the product managers previously classified.

Which solution should the ML engineer apply to perform multi-class text classification of the user feedback?

- Use the SageMaker Latent Dirichlet Allocation (LDA) algorithm.
- Use the SageMaker BlazingText algorithm. ✓
- Use the SageMaker Neural Topic Model (NTM) algorithm.
- Use the SageMaker CatBoost algorithm.

Explanation:
The correct answer is **B. Use the SageMaker BlazingText algorithm.**
Here's why:

Multi-class Text Classification: The problem requires assigning user feedback into one of several predefined categories (user interface issues, performance issues, etc.). This is a classic multi-class text classification problem.
Supervised Learning: The ML engineer has access to a large, accurately labeled dataset of historical user feedback classified by product managers. This indicates a supervised learning approach is suitable.
BlazingText's Suitability: Amazon SageMaker BlazingText is an algorithm optimized for text classification and word representation learning. It is highly efficient and scalable, making it appropriate for handling large datasets and delivering accurate results. It supports both supervised (text classification) and unsupervised (word embedding) tasks.
https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext.html
Let's examine why the other options are less suitable:

A. SageMaker Latent Dirichlet Allocation (LDA) algorithm: LDA is an unsupervised learning algorithm used for topic modeling. It discovers abstract "topics" present in a collection of documents. It is not suitable for multi-class classification where labels are known beforehand.
C. SageMaker Neural Topic Model (NTM) algorithm: Similar to LDA, NTM is also an unsupervised learning algorithm for topic modeling. While it uses neural networks, it still focuses on discovering underlying topics, not classifying data into predefined categories.
D. SageMaker CatBoost algorithm: CatBoost is a gradient boosting algorithm primarily designed for structured data (tabular data with numerical and categorical features). While it can theoretically be used for text classification with appropriate feature engineering (like TF-IDF), it's not the most efficient or direct approach compared to BlazingText, which is specifically built for text. Additionally, using CatBoost on text data typically requires significant feature engineering to convert the text into a suitable numerical representation. BlazingText handles text data directly without the need for extensive preprocessing and feature engineering.

Therefore, BlazingText is the most appropriate SageMaker algorithm for this specific scenario due to its capabilities in supervised text classification, efficiency, and scalability with large datasets. It is the best option because it directly addresses the need to classify the user feedback into predefined categories using a supervised learning approach with labeled data.

Question: 285

Exam Heist

A digital media company wants to build a customer churn prediction model by using tabular data. The model should clearly indicate whether a customer will stop using the company's services. The company wants to clean the data because the data contains some empty fields, duplicate values, and rare values.

Which solution will meet these requirements with the LEAST development effort?

- Use SageMaker Canvas to automatically clean the data and to prepare a categorical model.
- Use SageMaker Data Wrangler to clean the data. Use the built-in SageMaker XGBoost algorithm to train a classification model. ✓
- Use SageMaker Canvas automatic data cleaning and preparation tools. Use the built-in SageMaker XGBoost algorithm to train a regression model.
- Use SageMaker Data Wrangler to clean the data. Use the SageMaker Autopilot to train a regression model

Explanation:

The correct answer is **B**, utilizing SageMaker Data Wrangler for data cleaning and the built-in SageMaker XGBoost algorithm for training a classification model. Here's why:

The problem requires a solution for cleaning tabular data with empty fields, duplicates, and rare values, followed by building a churn prediction model (which is a classification problem). The key consideration is minimizing development effort.

Why B is the best option:

SageMaker Data Wrangler: This is purpose-built for data preparation and cleaning tasks. It provides a visual interface and a wide range of built-in transformations to handle missing values, remove duplicates, and deal with rare values, significantly reducing manual coding.

Built-in SageMaker XGBoost: XGBoost is a powerful and efficient algorithm commonly used for classification tasks. Using the built-in version within SageMaker avoids the need for custom container creation and deployment, further reducing development effort.

Classification Focus: Churn prediction is inherently a classification problem (customer will churn or not). XGBoost is well-suited for this type of problem.

Why other options are less suitable:

A & C (SageMaker Canvas): SageMaker Canvas is designed for citizen data scientists and emphasizes no-code/low-code model building. While it offers automatic data cleaning, it might not provide the same level of granular control and customization over data cleaning as Data Wrangler, especially for handling specific issues like rare values. Also, option C suggests a regression model, which is incorrect for a churn prediction (classification) problem.

D (SageMaker Autopilot with Regression): While SageMaker Autopilot automates model building, including feature engineering and algorithm selection, it can be overkill for this scenario. Moreover, the suggestion of using a regression model is incorrect since churn is a classification problem. Also, Autopilot requires some time to explore different model options and is not always the fastest for a well-defined problem. And suggesting Regression when it's clearly a classification task makes this wrong.

In summary: SageMaker Data Wrangler offers the most efficient way to clean the data, and XGBoost provides a readily available and effective classification algorithm, minimizing development effort while addressing all problem requirements.

Supporting Links:

SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
SageMaker XGBoost: https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html
Classification vs Regression: https://towardsdatascience.com/regression-vs-classification-in-machine-learning-4118d333e703

Question: 286

Exam Heist

A data engineer is evaluating customer data in Amazon SageMaker Data Wrangler. The data engineer will use the customer data to create a new model to predict customer behavior.

The engineer needs to increase the model performance by checking for multicollinearity in the dataset.

Which steps can the data engineer take to accomplish this with the LEAST operational effort? (Choose two.)

- [ ] Use SageMaker Data Wrangler to refit and transform the dataset by applying one-hot encoding to category-based variables.
- [x] Use SageMaker Data Wrangler diagnostic visualization. Use principal components analysis (PCA) and singular value decomposition (SVD) to calculate singular values. ✓
- [ ] Use the SageMaker Data Wrangler Quick Model visualization to quickly evaluate the dataset and to produce importance scores for each feature.
- [x] Use the SageMaker Data Wrangler Min Max Scaler transform to normalize the data. ✓
- [ ] Use SageMaker Data Wrangler diagnostic visualization. Use least absolute shrinkage and selection operator (LASSO) to plot coefficient values from a LASSO model that is trained on the dataset.

Explanation:

Let's analyze why options B and D are the correct choices for detecting multicollinearity with the least operational effort within SageMaker Data Wrangler.

Option B: Using SageMaker Data Wrangler's diagnostic visualization with Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) is a direct and efficient method to check for multicollinearity. PCA and SVD are dimensionality reduction techniques. When strong multicollinearity exists, some singular values will be close to zero. This indicates that the dataset's dimensions are highly correlated, implying multicollinearity. Data Wrangler's visual interface makes this analysis easier than coding it from scratch. This approach allows the data engineer to readily identify highly correlated features by examining the singular values.

Option D: While not directly detecting multicollinearity, using SageMaker Data Wrangler's MinMax Scaler transform to normalize the data is a crucial *pre-processing* step. Multicollinearity can sometimes be masked or exacerbated by differing scales of the independent variables. By scaling the features to a standard range (e.g., between 0 and 1), MinMax scaling ensures that no feature dominates the analysis solely based on its original scale. This standardization process enhances the accuracy of any multicollinearity detection method subsequently employed. Furthermore, many machine learning models perform better with normalized data, making this a beneficial operation regardless.

Why the other options are not optimal:

Option A: One-hot encoding converts categorical variables into numerical ones. While it's often necessary for model training, it doesn't directly address multicollinearity. It can even *increase* multicollinearity if categorical variables are highly correlated.

Option C: Quick Model in Data Wrangler focuses on feature importance based on a single model. While feature importance can *indirectly* suggest potential multicollinearity (e.g., several features exhibiting similar importance and high correlation), it's not a primary method for detection and relies on model-specific interpretations. Also, it requires training a model, making it more operationally expensive than direct diagnostics.

Option E: While LASSO (Least Absolute Shrinkage and Selection Operator) *can* be used to identify multicollinearity (because coefficients of highly correlated features tend to be driven to zero), it's not the *least* effort approach within Data Wrangler. The analysis requires interpreting the coefficients of a trained LASSO model. Also, it is important to note that LASSO can reduce the coefficients for one of the correlated predictors but not necessarily all of them. PCA and SVD provide a more direct visualization within Data Wrangler to highlight the issue of multicollinearity.

In summary, Data Wrangler's PCA/SVD visualizations quickly reveal the presence of multicollinearity, and MinMax scaling is a beneficial pre-processing step that can improve both multicollinearity detection and model performance. Thus, choices B and D provide the least operational effort to achieve the stated goal.

Supporting Links:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Multicollinearity: https://www.statisticshowto.com/multicollinearity/
PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
MinMax Scaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Question: 287

Exam Heist

A company processes millions of orders every day. The company uses Amazon DynamoDB tables to store order information. When customers submit new orders, the new orders are immediately added to the DynamoDB tables. New orders arrive in the DynamoDB tables continuously.

A data scientist must build a peak-time prediction solution. The data scientist must also create an Amazon QuickSight dashboard to display near real-time order insights. The data scientist needs to build a solution that will give QuickSight access to the data as soon as new order information arrives.

Which solution will meet these requirements with the LEAST delay between when a new order is processed and when QuickSight can access the new order information?

- ○ Use AWS Glue to export the data from Amazon DynamoDB to Amazon S3. Configure QuickSight to access the data in Amazon S3.

- Use Amazon Kinesis Data Streams to export the data from Amazon DynamoDB to Amazon S3. Configure QuickSight to access the data in Amazon S3.
- Use an API call from QuickSight to access the data that is in Amazon DynamoDB directly.
- Use Amazon Kinesis Data Firehose to export the data from Amazon DynamoDB to Amazon S3. Configure QuickSight to access the data in Amazon S3. ✓

Explanation:

The correct answer is **D. Use Amazon Kinesis Data Firehose to export the data from Amazon DynamoDB to Amazon S3. Configure QuickSight to access the data in Amazon S3.**

Here's a detailed justification:

The primary requirement is minimizing the delay between order processing and data availability in QuickSight. Direct access to DynamoDB from QuickSight is generally discouraged due to potential performance impacts on the database and limitations in QuickSight's ability to handle high-volume, near real-time updates efficiently (eliminating option C). Therefore, a streaming solution into a data lake like S3 is preferable.

Options A, B, and D all involve exporting data to S3 and then configuring QuickSight to access it. The key differentiator lies in the speed and efficiency of data delivery. AWS Glue (option A) is primarily an ETL service and not optimized for real-time data streaming. It's more suited for batch processing and schema discovery, introducing significant delays.

Kinesis Data Streams (option B) provides real-time data streaming capabilities but requires additional development effort to consume and load data into S3. You need to write custom consumer applications, adding latency and complexity.

Kinesis Data Firehose (option D) is specifically designed for delivering streaming data to data lakes like S3 with minimal configuration and near real-time latency. It automatically handles buffering, batching, compression, and encryption of data before delivery. This minimizes delay and simplifies the data pipeline. Firehose also provides built-in retry mechanisms and error handling, increasing the robustness of the solution. By directly piping data into S3, QuickSight can access it relatively quickly compared to the other options after Firehose buffer configuration settings.

Therefore, Kinesis Data Firehose offers the *least* delay between order processing and data availability in QuickSight by providing a managed, near real-time data streaming solution to S3, fulfilling the near real-time dashboarding requirement in QuickSight.

Relevant Documentation:

Amazon Kinesis Data Firehose: https://aws.amazon.com/kinesis/data-firehose/
Amazon QuickSight: https://aws.amazon.com/quicksight/
Amazon DynamoDB: https://aws.amazon.com/dynamodb/

Question: 288

Exam Heist

A data engineer is preparing a dataset that a retail company will use to predict the number of visitors to stores. The data engineer created an Amazon S3 bucket. The engineer subscribed the S3 bucket to an AWS Data Exchange data product for general economic indicators. The data engineer wants to join the economic indicator data to an existing table in Amazon Athena to merge with the business data. All these transformations must finish running in 30-60 minutes.

Which solution will meet these requirements MOST cost-effectively?

- Configure the AWS Data Exchange product as a producer for an Amazon Kinesis data stream. Use an Amazon Kinesis Data Firehose delivery stream to transfer the data to Amazon S3. Run an AWS Glue job that will merge the existing business data with the Athena table. Write the result set back to Amazon S3.
- Use an S3 event on the AWS Data Exchange S3 bucket to invoke an AWS Lambda function. Program the Lambda function to use Amazon SageMaker Data Wrangler to merge the existing business data with the Athena table. Write the result set back to Amazon S3.
- Use an S3 event on the AWS Data Exchange S3 bucket to invoke an AWS Lambda function. Program the Lambda function to run an AWS Glue job that will merge the existing business data with the Athena table. Write the results back to Amazon S3. ✓
- Provision an Amazon Redshift cluster. Subscribe to the AWS Data Exchange product and use the product to create an Amazon Redshift table. Merge the data in Amazon Redshift. Write the results back to Amazon S3.

Explanation:

Here's a detailed justification for why option C is the most cost-effective solution:

Option C leverages several AWS services efficiently for data integration. An S3 event trigger invoking a Lambda function provides an event-driven, serverless approach, eliminating the need for constantly running infrastructure. The Lambda function orchestrates the data transformation. AWS Glue is a fully managed ETL (Extract, Transform, Load) service specifically designed for data cataloging and transformation, ideal for merging data from different sources into a table accessible by

Athena. Lambda's ephemeral nature coupled with Glue's pay-as-you-go pricing creates a cost-effective solution as resources are only used when the Data Exchange product updates the S3 bucket. Finally writing the results back to S3 ensures data persistence.
Option A introduces Kinesis, which is better suited for streaming data. Given the data product likely delivers updates in batches, Kinesis adds unnecessary complexity and cost for this scenario. Option B suggests using SageMaker Data Wrangler within Lambda. Data Wrangler is designed for interactive data exploration and preparation, primarily for ML model training, making it overkill and less efficient than Glue for this data merging task. Option D involves provisioning a whole Redshift cluster, which is expensive and unnecessary for a task that can be accomplished with a serverless approach. Redshift is more suited for large-scale data warehousing and complex analytical queries.
Therefore, using Lambda to trigger an AWS Glue job to merge the data and write the output back to S3 is the most cost-effective approach due to the serverless nature of Lambda and the purpose-built ETL capabilities of Glue. The entire process is well-suited to completing within the required timeframe.
Relevant links for further research:

AWS Lambda: https://aws.amazon.com/lambda/
AWS Glue: https://aws.amazon.com/glue/
Amazon S3: https://aws.amazon.com/s3/
AWS Data Exchange: https://aws.amazon.com/data-exchange/

Question: 289

Exam Heist

A company operates large cranes at a busy port The company plans to use machine learning (ML) for predictive maintenance of the cranes to avoid unexpected breakdowns and to improve productivity.

The company already uses sensor data from each crane to monitor the health of the cranes in real time. The sensor data includes rotation speed, tension, energy consumption, vibration, pressure, and temperature for each crane. The company contracts AWS ML experts to implement an ML solution.

Which potential findings would indicate that an ML-based solution is suitable for this scenario? (Choose two.)

- [ ] The historical sensor data does not include a significant number of data points and attributes for certain time periods.
- [ ] The historical sensor data shows that simple rule-based thresholds can predict crane failures.
- [ ] The historical sensor data contains failure data for only one type of crane model that is in operation and lacks failure data of most other types of crane that are in operation.
- [x] The historical sensor data from the cranes are available with high granularity for the last 3 years. ✓
- [x] The historical sensor data contains most common types of crane failures that the company wants to predict. ✓

Explanation:
Here's a detailed justification of why options D and E are the most suitable indicators that an ML-based solution is appropriate for predictive maintenance of the cranes:

Option D: The historical sensor data from the cranes are available with high granularity for the last 3 years. A large, granular dataset spanning multiple years is crucial for training robust and accurate machine learning models. Machine learning algorithms thrive on data to learn complex patterns and relationships. Three years of data provide sufficient temporal depth to capture seasonality, trends, and cyclical patterns in crane behavior and performance. High granularity ensures that subtle changes preceding failures are captured, enabling the model to learn early warning signs. Without sufficient data volume and detail, the ML model would be prone to overfitting (memorizing the training data rather than generalizing) or underfitting (failing to capture the underlying patterns).
Option E: The historical sensor data contains most common types of crane failures that the company wants to predict. The success of a predictive maintenance model hinges on its ability to learn from past failures. If the historical data covers the predominant failure modes, the model has a higher chance of accurately identifying precursors and predicting their occurrence. A comprehensive failure dataset provides a diverse set of scenarios for the model to learn from, leading to better generalization and more reliable predictions in real-world operations. Lack of data on common failure modes would limit the model's ability to predict and prevent those specific types of breakdowns.
Why the other options are not ideal:

Option A: A lack of data is detrimental to ML model training.
Option B: If simple rules can predict failures, a complex ML solution might be unnecessary (although ML could still be used to automate the application of the rules).
Option C: Limited data on only one type of crane model severely restricts the model's ability to generalize to other crane types.
Supporting Cloud Computing Concepts:

Data Lakes: Storing the large volume of sensor data would typically involve a data lake solution on AWS, such as Amazon S3, to handle the variety and volume of the data.
Data Preprocessing: Before training the ML model, the sensor data needs to be cleaned, transformed, and potentially augmented, often using services like AWS Glue.
Machine Learning Services: AWS SageMaker provides the tools to build, train, and deploy ML models for predictive maintenance.
Authoritative Links for further research:

Amazon SageMaker: https://aws.amazon.com/sagemaker/
AWS Machine Learning: https://aws.amazon.com/machine-learning/
Predictive Maintenance with Machine Learning: (Search AWS documentation for "predictive maintenance" to find use cases and solutions)

Question: 290

Exam Heist

A company wants to create an artificial intelligence (AШ) yoga instructor that can lead large classes of students. The company needs to create a feature that can accurately count the number of students who are in a class. The company also needs a feature that can differentiate students who are performing a yoga stretch correctly from students who are performing a stretch incorrectly.

Determine whether students are performing a stretch correctly, the solution needs to measure the location and angle of each student's arms and legs. A data scientist must use Amazon SageMaker to access video footage of a yoga class by extracting image frames and applying computer vision models.

Which combination of models will meet these requirements with the LEAST effort? (Choose two.)

- [ ] Image Classification
- [ ] Optical Character Recognition (OCR)
- [x] Object Detection ✓
- [x] Pose estimation ✓
- [ ] Image Generative Adversarial Networks (GANs)

Explanation:
The correct answer is C and D: Object Detection and Pose Estimation.
Here's why:

Object Detection (C): The primary requirement is to count the number of students. Object detection models are designed to identify and locate specific objects within an image. In this scenario, the model would be trained to identify "students" and draw bounding boxes around each person in the image frame. This directly addresses the need for counting students in the class. Example Amazon SageMaker Object Detection algorithms include SSD, Faster R-CNN, and Yolo. (https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection.html)
Pose Estimation (D): The second requirement involves determining whether students are performing yoga stretches correctly, which requires understanding their body positioning. Pose estimation models are specifically designed to identify and track the key points (joints) of a person's body within an image. By analyzing the coordinates of these key points (e.g., elbows, knees, wrists), the model can calculate angles and distances between body parts, thereby determining the correctness of a pose. A pose estimation model outputs the (x, y) coordinates of the key points, which can be used to measure angles. Common Pose Estimation models can be found in popular ML frameworks such as TensorFlow. (https://paperswithcode.com/task/pose-estimation)
Here's why the other options are incorrect:

Image Classification (A): Image classification assigns a single label to an entire image (e.g., "cat," "dog," "yoga class"). It doesn't provide the ability to locate and count individual students or analyze their poses.
Optical Character Recognition (OCR) (B): OCR is used to extract text from images. It is irrelevant to counting students or analyzing their body positions.
Image Generative Adversarial Networks (GANs) (E): GANs are used to generate new, synthetic images that are similar to the training data. GANs are not relevant for this use case.

Question: 291

Exam Heist

An ecommerce company has used Amazon SageMaker to deploy a factorization machines (FM) model to suggest products for customers. The company's data science team has developed two new models by using the TensorFlow and PyTorch deep learning frameworks. The company needs

to use A/B testing to evaluate the new models against the deployed model.

The required A/B testing setup is as follows:

•Send 70% of traffic to the FM model, 15% of traffic to the TensorFlow model, and 15% of traffic to the PyTorch model.
•For customers who are from Europe, send all traffic to the TensorFlow model.

Which architecture can the company use to implement the required A/B testing setup?

- Create two new SageMaker endpoints for the TensorFlow and PyTorch models in addition to the existing SageMaker endpoint. Create an Application Load Balancer. Create a target group for each endpoint. Configure listener rules and add weight to the target groups. To send traffic to the TensorFlow model for customers who are from Europe, create an additional listener rule to forward traffic to the TensorFlow target group.
- Create two production variants for the TensorFlow and PyTorch models. Create an auto scaling policy and configure the desired A/B weights to direct traffic to each production variant. Update the existing SageMaker endpoint with the auto scaling policy. To send traffic to the TensorFlow model for customers who are from Europe, set the TargetVariant header in the request to point to the variant name of the TensorFlow model.
- Create two new SageMaker endpoints for the TensorFlow and PyTorch models in addition to the existing SageMaker endpoint. Create a Network Load Balancer. Create a target group for each endpoint. Configure listener rules and add weight to the target groups. To send traffic to the TensorFlow model for customers who are from Europe, create an additional listener rule to forward traffic to the TensorFlow target group.
- Create two production variants for the TensorFlow and PyTorch models. Specify the weight for each production variant in the SageMaker endpoint configuration. Update the existing SageMaker endpoint with the new configuration. To send traffic to the TensorFlow model for customers who are from Europe, set the TargetVariant header in the request to point to the variant name of the TensorFlow model. ✓

Explanation:
Here's a breakdown of why option D is the correct solution, along with supporting concepts and links:
Justification
Option D offers the most straightforward and efficient way to implement the A/B testing setup within the SageMaker environment. SageMaker endpoints have built-in capabilities for A/B testing using production variants and traffic distribution weights. This eliminates the need for external load balancers.
Production Variants: SageMaker endpoints support multiple production variants, each representing a different model version (in this case, the TensorFlow and PyTorch models).
Traffic Distribution: By configuring the weights for each production variant in the SageMaker endpoint configuration, you can directly control the percentage of traffic routed to each model (70% to FM, 15% to TensorFlow, 15% to PyTorch). This satisfies the primary A/B testing requirement.
Conditional Routing with `TargetVariant`: To handle the special case of routing all European traffic to the TensorFlow model, you can leverage the `TargetVariant` header in the inference request. This header allows you to override the default traffic distribution and explicitly direct a specific request to a particular production variant. The application making the inference request would need to determine the customer's location and set the `TargetVariant` header accordingly when the customer is from Europe.

Why Other Options Are Less Suitable:

A & C (Load Balancers): While using an Application Load Balancer (ALB) or Network Load Balancer (NLB) is a valid approach for routing traffic, it adds unnecessary complexity. SageMaker endpoints already provide built-in A/B testing capabilities, making external load balancers redundant for this specific scenario. Configuring and managing load balancers introduces overhead and increases operational costs.
B (Auto Scaling Policy): While auto scaling can be related to traffic management, it does not inherently fulfill the request for conditional routing based on geographic region. Configuring an auto scaling policy alongside production variants can be a good practice, but is not sufficient on its own to fulfill the requirements for A/B testing.
Key Concepts:

Amazon SageMaker Endpoints: Hosted model deployments that can serve real-time inference requests.
Production Variants: Different versions of a model deployed within the same SageMaker endpoint.
Traffic Distribution Weights: The percentage of traffic routed to each production variant.
`TargetVariant` **Header:** A request header that allows you to override the default traffic distribution and direct a request to a specific production variant.
Authoritative Links:

Deploy a Model to Amazon SageMaker Hosting Services: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
Invoke Endpoint: https://docs.aws.amazon.com/sagemaker/latest/dg/API_Runtime_InvokeEndpoint.html
Production Variants: https://docs.aws.amazon.com/sagemaker/latest/dg/production-variants.html

In summary, option D provides the most efficient, cost-effective, and SageMaker-native solution to implement the required A/B testing setup, leveraging production variants and the `TargetVariant` header for conditional routing.

Question: 292

Exam Heist

A data scientist stores financial datasets in Amazon S3. The data scientist uses Amazon Athena to query the datasets by using SQL.

The data scientist uses Amazon SageMaker to deploy a machine learning (ML) model. The data scientist wants to obtain inferences from the model at the SageMaker endpoint. However, when the data scientist attempts to invoke the SageMaker endpoint, the data scientist receives SQL statement failures. The data scientist's IAM user is currently unable to invoke the SageMaker endpoint.

Which combination of actions will give the data scientist's IAM user the ability to invoke the SageMaker endpoint? (Choose three.)

- [ ] Attach the AmazonAthenaFullAccess AWS managed policy to the user identity.
- [x] Include a policy statement for the data scientist's IAM user that allows the IAM user to perform the sagemaker:InvokeEndpoint action. ✓
- [x] Include an inline policy for the data scientist's IAM user that allows SageMaker to read S3 objects. ✓
- [ ] Include a policy statement for the data scientist's IAM user that allows the IAM user to perform the sagemaker:GetRecord action.
- [x] Include the SQL statement "USING EXTERNAL FUNCTION ml_function_name" in the Athena SQL query. ✓
- [ ] Perform a user remapping in SageMaker to map the IAM user to another IAM user that is on the hosted endpoint.

Explanation:
The data scientist needs to invoke a SageMaker endpoint from Athena using SQL, but is currently unable to do so due to IAM permissions. Here's a breakdown of why options B, C, and E are correct:

B. Include a policy statement for the data scientist's IAM user that allows the IAM user to perform the `sagemaker:InvokeEndpoint` action. This is fundamental. To call any SageMaker endpoint, the IAM user *must* have explicit permission to do so via the `sagemaker:InvokeEndpoint` action in an IAM policy. Without this permission, the call will be denied. https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-roles.html explains SageMaker roles and permissions.
C. Include an inline policy for the data scientist's IAM user that allows SageMaker to read S3 objects. Athena will invoke a SageMaker endpoint, which in turn might need to read data from S3 to make predictions. The IAM role used by the data scientist, or by SageMaker on behalf of the user, needs permission to access the relevant S3 buckets and objects. If SageMaker needs to access S3 for input data, it needs that permission delegated through the user's IAM role/policy. While the question doesn't explicitly mention SageMaker needing to read from S3, it's a very common scenario, and the inability to read S3 objects would cause the endpoint invocation to fail.
E. Include the SQL statement "USING EXTERNAL FUNCTION ml_function_name" in the Athena SQL query. To call a SageMaker endpoint from Athena, you need to define and use a user-defined function (UDF) that encapsulates the call to the endpoint. The `USING EXTERNAL FUNCTION` clause tells Athena to use an external function for prediction, which triggers the connection to the Sagemaker Endpoint. The `ml_function_name` represents a Lambda function that interacts with the endpoint.
Options A, D, and F are incorrect:

A. Attach the AmazonAthenaFullAccess AWS managed policy to the user identity. `AmazonAthenaFullAccess` provides full access to Athena services, but does not inherently grant the ability to invoke SageMaker endpoints. While it might grant some access necessary for setting up the Athena-SageMaker integration, it doesn't cover the endpoint invocation itself.
D. Include a policy statement for the data scientist's IAM user that allows the IAM user to perform the `sagemaker:GetRecord` action. The `sagemaker:GetRecord` action is used to retrieve ground truth data or annotations associated with a SageMaker labeling job, not for invoking an endpoint for inference.
F. Perform a user remapping in SageMaker to map the IAM user to another IAM user that is on the hosted endpoint. User remapping isn't a standard SageMaker feature. The data scientist's IAM user must have direct permission to invoke the endpoint; remapping to another user doesn't solve the root cause of the permission error.

Question: 293

Exam Heist

A data scientist is building a linear regression model. The scientist inspects the dataset and notices that the mode of the distribution is lower than the median, and the median is lower than the mean.

Which data transformation will give the data scientist the ability to apply a linear regression model?

- ( ) Exponential transformation
- ( ) Logarithmic transformation ✓

- Polynomial transformation
- Sinusoidal transformation

Explanation:
The question describes a dataset with a distribution where the mode < median < mean. This indicates a right-skewed or positively skewed distribution. Linear regression models assume that the relationship between variables is linear and that the residuals (the difference between predicted and actual values) are normally distributed. A right-skewed distribution violates the assumption of normality.
To address this skewness and make the data more suitable for linear regression, a logarithmic transformation is often applied. A logarithmic transformation compresses the higher values in the dataset more than the lower values, effectively pulling the tail of the distribution towards the center and reducing the skewness. This makes the distribution more symmetrical, bringing it closer to a normal distribution.
An exponential transformation would exacerbate the right skew, making the situation worse. Polynomial and sinusoidal transformations might introduce non-linearity that is not desired or appropriate for a linear regression model aiming to find linear relationships. Logarithmic transformations are therefore frequently used to transform skewed data into a more normal distribution, enabling the use of linear regression models more effectively.
Therefore, a logarithmic transformation is the best choice to address right skewness and enable the application of a linear regression model.
Further research:

Box-Cox Transformation: Explore more about data transformation for normality. https://www.statisticshowto.com/box-cox-transformation/
Skewness and Kurtosis: Understand the characteristics of data distribution. https://www.bmj.com/content/343/bmj.d230

Question: 294

Exam Heist

A data scientist receives a collection of insurance claim records. Each record includes a claim ID. the final outcome of the insurance claim, and the date of the final outcome.

The final outcome of each claim is a selection from among 200 outcome categories. Some claim records include only partial information. However, incomplete claim records include only 3 or 4 outcome categories from among the 200 available outcome categories. The collection includes hundreds of records for each outcome category. The records are from the previous 3 years.

The data scientist must create a solution to predict the number of claims that will be in each outcome category every month, several months in advance.

Which solution will meet these requirements?

- Perform classification every month by using supervised learning of the 200 outcome categories based on claim contents.
- Perform reinforcement learning by using claim IDs and dates. Instruct the insurance agents who submit the claim records to estimate the expected number of claims in each outcome category every month.
- Perform forecasting by using claim IDs and dates to identify the expected number of claims in each outcome category every month. ✓
- Perform classification by using supervised learning of the outcome categories for which partial information on claim contents is provided. Perform forecasting by using claim IDs and dates for all other outcome categories.

Explanation:
The correct answer is **C. Perform forecasting by using claim IDs and dates to identify the expected number of claims in each outcome category every month.**
Here's a detailed justification:

Forecasting Focus: The core requirement is to predict the *number* of claims per outcome category over time (monthly, several months in advance). This directly aligns with the purpose of forecasting. Forecasting techniques are designed to predict future values based on historical time series data. In this case, the time series data is the count of claims per outcome category over the past 3 years.
Time Series Data: Claim IDs and dates constitute a time series. Each claim ID can be associated with a date, and by aggregating claims based on their outcome category and date, you create a time series of claim counts per category. Forecasting algorithms can then analyze these time series to predict future claim counts.
Suitability for Numerous Categories: The fact that there are 200 outcome categories doesn't preclude forecasting. You can perform separate forecasting exercises for each outcome category. While computationally more intensive than a single model, it's a valid and potentially more accurate approach.
Why other options are wrong:

A (Classification): Classification aims to predict the *category* of a new instance (a new claim). While it could be used on individual claims to predict their outcome, it doesn't directly address the requirement of predicting the *number* of claims *per category* over time.
B (Reinforcement Learning): Reinforcement learning requires an environment where an agent takes actions and receives rewards or penalties. While conceptually possible, it's an overkill for this problem. It would be far more complex to implement and less direct than forecasting. Relying on insurance agents' estimates makes the solution subjective and potentially inaccurate, and it doesn't leverage the existing historical data.
D (Hybrid Classification/Forecasting): This option is unnecessarily complex. While there's a mention of "partial information," the problem description also states, "The collection includes hundreds of records for each outcome category." Therefore, there is sufficient historical data to perform forecasting for all 200 categories. Using classification for some categories based on incomplete data while using forecasting for the rest is not a clean or efficient solution. The problem statement indicates that the incomplete records still allow for identifying their actual outcome category.
In summary, forecasting provides the most direct and appropriate approach to predict the future number of claims in each outcome category based on historical claim data and dates, aligning perfectly with the problem's requirements.
Supporting Links:

AWS Time Series Forecasting: https://aws.amazon.com/forecast/
Amazon Forecast Documentation: https://docs.aws.amazon.com/forecast/latest/dg/what-is-forecast.html
Time Series Analysis: https://en.wikipedia.org/wiki/Time_series

Question: 295

Exam Heist

A retail company stores 100 GB of daily transactional data in Amazon S3 at periodic intervals. The company wants to identify the schema of the transactional data. The company also wants to perform transformations on the transactional data that is in Amazon S3.

The company wants to use a machine learning (ML) approach to detect fraud in the transformed data.

Which combination of solutions will meet these requirements with the LEAST operational overhead? (Choose three.)

- [ ] Use Amazon Athena to scan the data and identify the schema.
- [x] Use AWS Glue crawlers to scan the data and identify the schema. ✓
- [ ] Use Amazon Redshift to store procedures to perform data transformations.
- [x] Use AWS Glue workflows and AWS Glue jobs to perform data transformations. ✓
- [ ] Use Amazon Redshift ML to train a model to detect fraud.
- [x] Use Amazon Fraud Detector to train a model to detect fraud. ✓

Explanation:
The best combination of solutions to meet the retail company's needs with the least operational overhead is **B, D, and F**. Here's why:

B. Use AWS Glue crawlers to scan the data and identify the schema: AWS Glue crawlers automatically infer the schema of data stored in Amazon S3. They connect to the data stores, crawl through the data, extract schema information, and store it in the AWS Glue Data Catalog. This is a serverless and automated way to discover the schema of the transactional data, minimizing manual effort. https://docs.aws.amazon.com/glue/latest/dg/add-crawler.html
D. Use AWS Glue workflows and AWS Glue jobs to perform data transformations: AWS Glue provides a fully managed ETL (extract, transform, load) service. AWS Glue workflows can orchestrate multiple AWS Glue jobs that perform the necessary data transformations on the transactional data stored in S3. AWS Glue's serverless nature eliminates the need for managing infrastructure, reducing operational overhead. Using AWS Glue jobs and workflows provides a scalable and cost-effective way to prepare data for fraud detection. https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html
F. Use Amazon Fraud Detector to train a model to detect fraud: Amazon Fraud Detector is a fully managed service specifically designed to detect fraudulent activities. It uses machine learning to identify potential fraud in real-time. By leveraging Amazon Fraud Detector, the company can quickly and easily build and deploy a fraud detection model without needing to have deep machine learning expertise. This is a much simpler approach than building and managing a custom ML model using Amazon Redshift ML. https://aws.amazon.com/fraud-detector/
Here's why the other options are less ideal:

A. Use Amazon Athena to scan the data and identify the schema: While Athena can scan data in S3, it requires knowledge of the schema beforehand, or manual schema definition. Using AWS Glue crawlers offers automatic schema discovery, making it a more suitable option in this scenario.

C. Use Amazon Redshift stored procedures to perform data transformations: Using Redshift stored procedures for data transformations can be costly for large datasets. Glue is generally better suited and optimized for ETL processing of large datasets stored in S3.
E. Use Amazon Redshift ML to train a model to detect fraud: While Redshift ML allows you to create, train, and deploy Amazon SageMaker models using SQL, it requires more setup and manual effort compared to using Amazon Fraud Detector, which is a purpose-built fraud detection service. Fraud Detector provides pre-built fraud detection algorithms and simplifies the model building and deployment process.
By choosing AWS Glue crawlers for schema discovery, AWS Glue for data transformations, and Amazon Fraud Detector for fraud detection, the retail company can achieve its goals with minimal operational overhead and leverage the benefits of serverless and managed AWS services.

Question: 296

Exam Heist

A data scientist uses Amazon SageMaker Data Wrangler to define and perform transformations and feature engineering on historical data. The data scientist saves the transformations to SageMaker Feature Store.

The historical data is periodically uploaded to an Amazon S3 bucket. The data scientist needs to transform the new historic data and add it to the online feature store. The data scientist needs to prepare the new historic data for training and inference by using native integrations.

Which solution will meet these requirements with the LEAST development effort?

- Use AWS Lambda to run a predefined SageMaker pipeline to perform the transformations on each new dataset that arrives in the S3 bucket.
- Run an AWS Step Functions step and a predefined SageMaker pipeline to perform the transformations on each new dataset that arrives in the S3 bucket.
- Use Apache Airflow to orchestrate a set of predefined transformations on each new dataset that arrives in the S3 bucket.
- Configure Amazon EventBridge to run a predefined SageMaker pipeline to perform the transformations when a new data is detected in the S3 bucket. ✓

Explanation:
Here's a detailed justification for why option D is the best solution:
The problem requires automated transformation of new historical data arriving in an S3 bucket and its integration with the SageMaker Feature Store, minimizing development effort. Option D, using Amazon EventBridge to trigger a predefined SageMaker pipeline upon new data detection in S3, is the most efficient approach for several reasons.
EventBridge excels at reacting to events within AWS services. Configuring it to listen for S3 object creation events and triggering a SageMaker pipeline directly addresses the need for automated transformation when new data is uploaded. Because the data scientist has already defined the transformations within a SageMaker pipeline using Data Wrangler, EventBridge simply acts as the trigger, minimizing the need for custom code. This "event-driven architecture" reduces operational overhead.
AWS Lambda (option A), while capable of processing data, requires writing custom functions to handle data retrieval from S3 and triggering the SageMaker pipeline. Step Functions (option B) adds unnecessary complexity for this simple trigger-and-execute workflow. Apache Airflow (option C) is an excellent orchestrator for complex workflows, but it's overkill for the given scenario and introduces additional infrastructure management. Airflow necessitates managing the Airflow infrastructure itself.
The key advantage of EventBridge is its serverless nature and tight integration with AWS services like S3 and SageMaker. It eliminates the need to manage servers or write extensive custom code for data retrieval and orchestration. The SageMaker pipeline is already defined, further reducing development effort. This makes EventBridge the most straightforward and cost-effective way to automatically process and integrate new data with the Feature Store.
References:

Amazon EventBridge: https://aws.amazon.com/eventbridge/
Amazon SageMaker Pipelines: https://aws.amazon.com/sagemaker/pipelines/
Amazon S3 Event Notifications: https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventNotifications.html

Question: 297

Exam Heist

An insurance company developed a new experimental machine learning (ML) model to replace an existing model that is in production. The company must validate the quality of predictions from the new experimental model in a production environment before the company uses the new experimental model to serve general user requests.

New one model can serve user requests at a time. The company must measure the performance of the new experimental model without affecting the current live traffic.

Which solution will meet these requirements?

- A/B testing
- Canary release
- Shadow deployment ✓
- Blue/green deployment

Explanation:

The correct answer is C, Shadow Deployment. Here's a detailed justification:

Shadow deployment is the optimal strategy because it allows the new model to receive the same live traffic as the existing production model *without* actually serving those requests. This parallel processing is key. The experimental model's predictions are captured and analyzed alongside the production model's output, enabling a direct comparison of performance metrics like accuracy, latency, and error rates in a real-world setting. Critically, the shadow model's output does not influence the user experience or the live traffic flow; the production model continues to handle all actual user requests. This minimizes risk during the evaluation phase.

A/B testing (A) involves splitting live traffic between different model versions, which directly impacts users and violates the requirement of not affecting current live traffic. Canary releases (B) incrementally expose a new model to a small subset of users, gradually increasing the load, again affecting live traffic and potentially user experience, albeit to a lesser extent than A/B testing. Blue/Green deployments (D) involve deploying the new version alongside the existing version and then switching traffic between them; this is generally done after initial testing and is not suited for parallel, non-influential evaluation.

Shadow deployment allows for rigorous assessment of the new model's performance under real-world load and data characteristics before it's actually put into service, ensuring that the transition is well-informed and minimizes the potential for negative impact on users. It enables performance metric data collection without impacting real-time responses.

For more information, research "shadow deployment strategies," "testing ML models in production," and "model deployment best practices." Resources like AWS documentation on deploying machine learning models with SageMaker, along with articles on MLOps and model monitoring, provide additional insight. AWS documentation specifically mentions shadow testing as an evaluation strategy.https://docs.aws.amazon.com/sagemaker/

Question: 298

Exam Heist

A company deployed a machine learning (ML) model on the company website to predict real estate prices. Several months after deployment, an ML engineer notices that the accuracy of the model has gradually decreased.

The ML engineer needs to improve the accuracy of the model. The engineer also needs to receive notifications for any future performance issues.

Which solution will meet these requirements?

- Perform incremental training to update the model. Activate Amazon SageMaker Model Monitor to detect model performance issues and to send notifications. ✓
- Use Amazon SageMaker Model Governance. Configure Model Governance to automatically adjust model hyperparameters. Create a performance threshold alarm in Amazon CloudWatch to send notifications.
- Use Amazon SageMaker Debugger with appropriate thresholds. Configure Debugger to send Amazon CloudWatch alarms to alert the team. Retrain the model by using only data from the previous several months.
- Use only data from the previous several months to perform incremental training to update the model. Use Amazon SageMaker Model Monitor to detect model performance issues and to send notifications.

Explanation:

The correct answer is **A. Perform incremental training to update the model. Activate Amazon SageMaker Model Monitor to detect model performance issues and to send notifications.**

Here's why:

Model Drift: The scenario describes model drift, where a model's performance degrades over time because the data it's predicting on changes. This is common in real-world ML deployments, especially in dynamic markets like real estate. https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html

Incremental Training: Incremental training (also known as online learning or continuous training) addresses model drift by updating the model with new data regularly. This allows the model to adapt to evolving patterns in the real estate market. It's more efficient than retraining from scratch each time.

Amazon SageMaker Model Monitor: This AWS service is specifically designed to detect and alert on model drift in real-time. It monitors the model's input and output data and compares them to a baseline (e.g., the data used to train the original model). When significant deviations occur, it triggers alerts. https://aws.amazon.com/sagemaker/model-monitor/

Notifications: SageMaker Model Monitor can be configured to send notifications through various channels, such as Amazon CloudWatch alarms, which can then trigger email alerts or other actions.

Why other options are not the best:

B. Amazon SageMaker Model Governance and Hyperparameter Tuning: While Model Governance helps manage model lifecycle, it's not primarily for real-time drift detection. Automatically adjusting hyperparameters might help, but doesn't directly address the underlying issue of changing data patterns and also model governance is not designed for this task. https://docs.aws.amazon.com/sagemaker/latest/dg/model-governance.html
C. Amazon SageMaker Debugger: Debugger focuses on identifying and fixing issues during model training (e.g., vanishing gradients). It's not for runtime drift detection. Retraining only on recent data can lead to overfitting on recent trends, potentially ignoring valuable long-term patterns. https://docs.aws.amazon.com/sagemaker/latest/dg/debugger.html
D. Using Only Recent Data and Model Monitor: Although this approach is close to the correct answer, incremental training is preferable because it leverages existing model knowledge and then gradually fine-tunes it with new data. Completely discarding historical data to retrain exclusively on recent data will lead to poor results. While Model Monitor is correctly configured to generate notification of performance issues.

Question: 299

Exam Heist

A university wants to develop a targeted recruitment strategy to increase new student enrollment. A data scientist gathers information about the academic performance history of students. The data scientist wants to use the data to build student profiles. The university will use the profiles to direct resources to recruit students who are likely to enroll in the university.

Which combination of steps should the data scientist take to predict whether a particular student applicant is likely to enroll in the university? (Choose two.)

- [x] Use Amazon SageMaker Ground Truth to sort the data into two groups named "enrolled" or "not enrolled." ✓
- [ ] Use a forecasting algorithm to run predictions.
- [ ] Use a regression algorithm to run predictions.
- [x] Use a classification algorithm to run predictions. ✓
- [ ] Use the built-in Amazon SageMaker k-means algorithm to cluster the data into two groups named "enrolled" or "not enrolled."

Explanation:
The correct answer is AD. Here's a detailed justification:
A. Use Amazon SageMaker Ground Truth to sort the data into two groups named "enrolled" or "not enrolled." - Correct
Amazon SageMaker Ground Truth allows you to build highly accurate training datasets quickly. In this scenario, Ground Truth can be used to label the historical student data into two distinct categories: "enrolled" and "not enrolled." This labeling process transforms the raw data into a supervised learning dataset. Since the university already has historical data on which students enrolled, Ground Truth would be ideal for structuring and labeling the data. It enables you to create a high-quality dataset which is essential for training a robust model.

Amazon SageMaker Ground Truth Documentation
D. Use a classification algorithm to run predictions. - Correct
This is a binary classification problem because the goal is to predict whether a student will enroll (yes/no, or 1/0). Classification algorithms are designed to predict the category a given data point belongs to. Algorithms such as logistic regression, decision trees, or support vector machines (SVM) would be appropriate choices for training a model to predict enrollment likelihood. These algorithms can learn the relationships between student characteristics and enrollment status from the labeled dataset created in step A.

Classification Algorithms in Machine Learning
Why other options are incorrect:

B. Use a forecasting algorithm to run predictions. Forecasting algorithms are generally used for predicting future values of a time series. While enrollment might have a temporal component, the primary problem here is classifying individual students based on their characteristics, not forecasting overall enrollment trends.
C. Use a regression algorithm to run predictions. Regression algorithms are used to predict continuous values, such as predicting a student's GPA or test score. Since the outcome variable (enrollment) is a binary categorical variable, regression is not appropriate.
E. Use the built-in Amazon SageMaker k-means algorithm to cluster the data into two groups named "enrolled" or "not enrolled." K-means is an unsupervised clustering algorithm. While it could potentially separate students into groups, it does so without leveraging existing "enrolled" or "not enrolled" labels. Because this is supervised learning scenario due to historical

enrollment data, classification is more accurate. K-means is better suited for scenarios with no ground truth labels.

Question: 300

Exam Heist

A machine learning (ML) specialist is using the Amazon SageMaker DeepAR forecasting algorithm to train a model on CPU-based Amazon EC2 On-Demand instances. The model currently takes multiple hours to train. The ML specialist wants to decrease the training time of the model.

Which approaches will meet this requirement? (Choose two.)

- [ ] Replace On-Demand Instances with Spot Instances.
- [ ] Configure model auto scaling dynamically to adjust the number of instances automatically.
- [x] Replace CPU-based EC2 instances with GPU-based EC2 instances. ✓
- [x] Use multiple training instances. ✓
- [ ] Use a pre-trained version of the model. Run incremental training.

Explanation:

The correct answer is **C and D**. Here's a breakdown of why:

C: Replace CPU-based EC2 instances with GPU-based EC2 instances. Deep learning models, like those used by DeepAR, often benefit significantly from GPU acceleration. GPUs are designed for parallel processing, making them well-suited for the matrix operations involved in neural network training. Replacing CPU instances with GPU instances can drastically reduce the training time. https://aws.amazon.com/ec2/instance-types/gpu/

D: Use multiple training instances. SageMaker supports distributed training, which involves splitting the training data and distributing the workload across multiple instances. This parallelism significantly reduces the overall training time. By distributing the training workload across multiple instances, the amount of time spent training on each instance is reduced, thereby reducing the overall time. https://docs.aws.amazon.com/sagemaker/latest/dg/distributed-training.html

Let's examine why the other options are not ideal:

A: Replace On-Demand Instances with Spot Instances. While Spot Instances can save costs, they can be interrupted if the Spot price exceeds your bid. This interruption can prolong the overall training time if the training job needs to be restarted. While cost-effective, it is not consistent for reducing overall time.

B: Configure model auto scaling dynamically to adjust the number of instances automatically. Auto Scaling for inference is useful for managing resources based on real-time traffic. However, for a fixed training job, Auto Scaling isn't directly applicable as the number of instances needed is determined beforehand or during the training setup (for distributed training).

E: Use a pre-trained version of the model. Run incremental training. Incremental training can reduce training time, but it requires a suitable pre-trained model and may not be applicable to all scenarios or datasets. Also, DeepAR doesn't explicitly support "incremental training" in the traditional transfer learning sense. The model needs to be initialized with weights from other models.

Question: 301

Exam Heist

A chemical company has developed several machine learning (ML) solutions to identify chemical process abnormalities. The time series values of independent variables and the labels are available for the past 2 years and are sufficient to accurately model the problem.

The regular operation label is marked as 0 The abnormal operation label is marked as 1. Process abnormalities have a significant negative effect on the company's profits. The company must avoid these abnormalities.

Which metrics will indicate an ML solution that will provide the GREATEST probability of detecting an abnormality?

- ( ) Precision = 0.91 -Recall = 0.6
- (x) Precision = 0.61 -Recall = 0.98 ✓
- ( ) Precision = 0.7 -Recall = 0.9
- ( ) Precision = 0.98 -Recall = 0.8

Explanation:

Here's a detailed justification for why option B is the best choice, considering the need to maximize the detection of abnormalities, even if it comes at the cost of some false positives:

The core objective is to minimize the risk of *missing* abnormalities, as they significantly impact company profits. This means we prioritize *recall*. Recall measures the ability of the model to find *all* the actual positive cases (abnormalities in this case). A high recall value indicates that the model is effective at identifying most of the true abnormalities.
Precision, on the other hand, measures the proportion of positive identifications that were actually correct. High precision means fewer false positives (instances predicted as abnormal that were actually normal). While minimizing false positives is desirable, it is secondary to maximizing the detection of true abnormalities in this scenario, where the cost of a false negative (missing an abnormality) is high.
Option B (Precision = 0.61, Recall = 0.98) offers the highest recall (0.98). This means the model will identify 98% of the actual process abnormalities. While the precision is lower (61%), indicating more false alarms, the benefit of catching nearly all true abnormalities outweighs the cost of investigating those false alarms.
Options A, C, and D have lower recall values, which means they miss a greater proportion of the actual abnormalities. Since the company wants to maximize the probability of detecting abnormalities to prevent profit loss, these options are less desirable. A high-precision, low-recall model might lull the company into a false sense of security by reporting few abnormalities, while actually missing many.
In summary, while a perfect model would have both high precision and high recall, in this specific business context, the company is willing to tolerate more false positives (lower precision) in order to minimize false negatives (maximize recall) and avoid costly process abnormalities. Therefore, prioritizing recall is critical.
For further research on precision and recall, refer to these resources:

Wikipedia: https://en.wikipedia.org/wiki/Precision_and_recall
Scikit-learn documentation: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

Question: 302

Exam Heist

An online delivery company wants to choose the fastest courier for each delivery at the moment an order is placed. The company wants to implement this feature for existing users and new users of its application. Data scientists have trained separate models with XGBoost for this purpose, and the models are stored in Amazon S3. There is one model for each city where the company operates.

Operation engineers are hosting these models in Amazon EC2 for responding to the web client requests, with one instance for each model, but the instances have only a 5% utilization in CPU and memory. The operation engineers want to avoid managing unnecessary resources.

Which solution will enable the company to achieve its goal with the LEAST operational overhead?

- Create an Amazon SageMaker notebook instance for pulling all the models from Amazon S3 using the boto3 library. Remove the existing instances and use the notebook to perform a SageMaker batch transform for performing inferences offline for all the possible users in all the cities. Store the results in different files in Amazon S3. Point the web client to the files.
- Prepare an Amazon SageMaker Docker container based on the open-source multi-model server. Remove the existing instances and create a multi-model endpoint in SageMaker instead, pointing to the S3 bucket containing all the models. Invoke the endpoint from the web client at runtime, specifying the TargetModel parameter according to the city of each request. ✓
- Keep only a single EC2 instance for hosting all the models. Install a model server in the instance and load each model by pulling it from Amazon S3. Integrate the instance with the web client using Amazon API Gateway for responding to the requests in real time, specifying the target resource according to the city of each request.
- Prepare a Docker container based on the prebuilt images in Amazon SageMaker. Replace the existing instances with separate SageMaker endpoints, one for each city where the company operates. Invoke the endpoints from the web client, specifying the URL and EndpointName parameter according to the city of each request.

Explanation:
Here's a detailed justification for why option B is the best solution, along with supporting concepts and links:
Why Option B is the Best Solution:
Option B leverages SageMaker's multi-model endpoint capability, which directly addresses the problem of underutilized EC2 instances. Instead of running one EC2 instance per city/model (resulting in 5% utilization), a single SageMaker endpoint can host all the models.

Efficiency: SageMaker optimizes resource utilization by dynamically loading and unloading models based on request patterns. This eliminates the overhead of keeping idle instances running.
Scalability: SageMaker automatically scales the endpoint based on traffic, ensuring the service can handle peak loads without manual intervention.
Ease of Management: SageMaker abstracts away the complexities of model deployment, serving, and scaling, reducing operational overhead. By using an open-source multi-model server within a Docker container in SageMaker, the operation team benefits from built-in features like model management, versioning, and monitoring.

Targeted Inference: The `TargetModel` parameter allows the web client to specify which city's model to use for each request, making the solution adaptable to user location.
Why Other Options are Suboptimal:

Option A: Batch inference is not suitable for the requirement of delivering the fastest courier *at the moment* an order is placed. It involves offline processing, making it unsuitable for real-time decisions.
Option C: While consolidating models onto a single EC2 instance is better than the original setup, it still requires manual management of the instance, model loading/unloading, and API Gateway integration. It lacks the auto-scaling and resource optimization benefits of SageMaker.
Option D: Creating separate SageMaker endpoints for each city defeats the purpose of reducing operational overhead. It's essentially the same as the original EC2 setup, but with SageMaker endpoints instead, multiplying the effort.
Supporting Concepts and Links:

Amazon SageMaker Multi-Model Endpoints: This is a crucial feature for hosting multiple models on a single endpoint. https://docs.aws.amazon.com/sagemaker/latest/dg/multi-model-endpoints.html
Amazon SageMaker Inference: Learn more about various inference options in SageMaker.https://aws.amazon.com/sagemaker/inference/
SageMaker Docker Containers: Information about building and using Docker containers with SageMaker. https://docs.aws.amazon.com/sagemaker/latest/dg/docker-container.html
In summary, Option B provides the most efficient, scalable, and manageable solution by leveraging SageMaker's multi-model endpoint capabilities. This approach minimizes operational overhead while meeting the real-time inference requirements of the delivery company.

Question: 303

Exam Heist

A company builds computer-vision models that use deep learning for the autonomous vehicle industry. A machine learning (ML) specialist uses an Amazon EC2 instance that has a CPU:GPU ratio of 12:1 to train the models.

The ML specialist examines the instance metric logs and notices that the GPU is idle half of the time. The ML specialist must reduce training costs without increasing the duration of the training jobs.

Which solution will meet these requirements?

- Switch to an instance type that has only CPUs.
- Use a heterogeneous cluster that has two different instances groups.
- Use memory-optimized EC2 Spot Instances for the training jobs.
- Switch to an instance type that has a CPU:GPU ratio of 6:1. ✓

Explanation:
The problem states that the GPU is idle 50% of the time, indicating that the current instance configuration is over-provisioned in terms of GPU capacity relative to the CPU workload. The goal is to reduce costs without increasing training duration.
Option A, switching to CPU-only instances, is incorrect because deep learning models for computer vision heavily rely on GPUs for efficient training. Removing the GPU will drastically increase the training time, violating the requirement.
Option B, using a heterogeneous cluster with two different instance groups, might introduce unnecessary complexity in managing the training process. While it could be a solution, it's not the most straightforward approach to address the CPU:GPU imbalance. Moreover, splitting the workload might add communication overhead, potentially impacting training time.
Option C, using memory-optimized EC2 Spot Instances, focuses on memory capacity and Spot Instances' cost-saving mechanism. However, the problem is not about memory constraints but an imbalance between CPU and GPU utilization. While Spot Instances can reduce costs, they don't address the core issue of the GPU being idle and are subject to interruption, which may increase the overall duration or complexity of training jobs due to the need to handle preemptions and restarts.
Option D, switching to an instance type with a CPU:GPU ratio of 6:1, directly addresses the problem of the GPU being underutilized. By selecting an instance type with fewer CPUs per GPU, the CPU can keep the GPU busy for a higher percentage of the time. Since the GPU is idle 50% of the time with the 12:1 ratio, halving the CPU count relative to the GPU (to 6:1) will more closely match the workload requirements, thus increasing GPU utilization and reducing the need for over-provisioned CPU resources. This would result in reduced instance costs without sacrificing training performance. This is the most efficient and direct solution.
Therefore, option D is the most appropriate solution.
https://aws.amazon.com/ec2/instance-types/https://aws.amazon.com/ec2/spot/

Question: 304

Exam Heist

A company wants to forecast the daily price of newly launched products based on 3 years of data for older product prices, sales, and rebates. The time-series data has irregular timestamps and is missing some values.

Data scientist must build a dataset to replace the missing values. The data scientist needs a solution that resamples the data daily and exports the data for further modeling.

Which solution will meet these requirements with the LEAST implementation effort?

- Use Amazon EMR Serverless with PySpark.
- Use AWS Glue DataBrew.
- Use Amazon SageMaker Studio Data Wrangler. ✓
- Use Amazon SageMaker Studio Notebook with Pandas.

Explanation:

The correct answer is C: Use Amazon SageMaker Studio Data Wrangler. Here's a detailed justification:
SageMaker Studio Data Wrangler excels at data preparation tasks, especially for machine learning projects. It offers a visual interface and pre-built data transformations, which significantly reduces implementation effort. Specifically, Data Wrangler's built-in time series capabilities directly address the irregular timestamps and missing values. It provides features for resampling time-series data to regular intervals (daily in this case) and imputing missing values using various methods (e.g., linear interpolation, mean, median). The visual interface allows the data scientist to easily define these transformations without writing extensive code. The resulting processed dataset can then be exported for further modeling.
Option A (Amazon EMR Serverless with PySpark) involves setting up and managing a distributed computing environment, writing PySpark code for data manipulation, and handling time series specific resampling. This requires considerably more effort compared to Data Wrangler's out-of-the-box features.
Option B (AWS Glue DataBrew) is a data preparation tool, but it's primarily designed for data integration and cleansing at scale. While it can handle some time-series operations, it doesn't offer the specialized time-series imputation and resampling features found in Data Wrangler, making implementation more complex.
Option D (Amazon SageMaker Studio Notebook with Pandas) requires the data scientist to write Python code using Pandas to handle resampling and missing value imputation. While Pandas is powerful, the need to write all the code from scratch increases the implementation effort compared to Data Wrangler's visual interface and pre-built transformations. Data Wrangler is purpose built for ML data prep and includes features like data visualizations and analysis that pandas alone doesn't provide in a managed fashion.
In summary, SageMaker Studio Data Wrangler provides a streamlined, visual, and specialized environment for time-series data preparation, directly addressing the requirements of resampling to daily intervals and imputing missing values with minimal coding effort.
Relevant Links:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
AWS Glue DataBrew: https://aws.amazon.com/glue/databrew/

Question: 305

Exam Heist

A data scientist is building a forecasting model for a retail company by using the most recent 5 years of sales records that are stored in a data warehouse. The dataset contains sales records for each of the company's stores across five commercial regions. The data scientist creates a working dataset with StoreID. Region. Date, and Sales Amount as columns. The data scientist wants to analyze yearly average sales for each region. The scientist also wants to compare how each region performed compared to average sales across all commercial regions.

Which visualization will help the data scientist better understand the data trend?

- Create an aggregated dataset by using the Pandas GroupBy function to get average sales for each year for each store. Create a bar plot, faceted by year, of average sales for each store. Add an extra bar in each facet to represent average sales.
- Create an aggregated dataset by using the Pandas GroupBy function to get average sales for each year for each store. Create a bar plot, colored by region and faceted by year, of average sales for each store. Add a horizontal line in each facet to represent average sales.
- Create an aggregated dataset by using the Pandas GroupBy function to get average sales for each year for each region. Create a bar plot of average sales for each region. Add an extra bar in each facet to represent average sales.
- Create an aggregated dataset by using the Pandas GroupBy function to get average sales for each year for each region. Create a bar plot, faceted by year, of average sales for each region. Add a horizontal line in each facet to represent average sales. ✓

Explanation:

The correct answer is **D**. Here's a breakdown of why:
The problem requires visualizing yearly regional sales trends compared to the overall average. Option D provides a clear and effective way to achieve this.
Aggregation: The first step is to aggregate the data to the appropriate level. Grouping by year and region using Pandas `GroupBy` provides the yearly average sales for each region, which is essential for the analysis.
Faceted Bar Plot: Creating a bar plot faceted by year is crucial. Faceting allows for side-by-side comparison of regional sales performance across different years, making it easy to identify trends and anomalies.
Horizontal Line for Overall Average: Adding a horizontal line to each facet representing the overall average sales across all regions for that year is a key element. This provides a benchmark against which to compare each region's performance. This visual cue instantly highlights whether a region outperformed or underperformed compared to the average.
Why other options are less ideal:

Option A & B: These are too granular, focusing on average sales for each store, and that is not useful for understanding the trend of Region sales compared to average. The question states that the data scientist wants to analyze yearly average sales for each region and to compare how each region performed compared to average sales across all commercial regions.
Option C: Not faceting by year makes it difficult to track sales trends across the years. This option only gives the average across all 5 years, losing the temporal component of the data.

In summary, option D's combination of regional aggregation, yearly faceting, and average baseline provides a clear and concise visualization that directly addresses the requirements of the problem.
Supporting Resources:

Pandas GroupBy: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html
Matplotlib Bar Plots: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html
Seaborn Faceting: https://seaborn.pydata.org/tutorial/axis_grids.html

Question: 306

Exam Heist

A company uses sensors on devices such as motor engines and factory machines to measure parameters, temperature and pressure. The company wants to use the sensor data to predict equipment malfunctions and reduce services outages.

Machine learning (ML) specialist needs to gather the sensors data to train a model to predict device malfunctions. The ML specialist must ensure that the data does not contain outliers before training the model.

How can the ML specialist meet these requirements with the LEAST operational overhead?

- Load the data into an Amazon SageMaker Studio notebook. Calculate the first and third quartile. Use a SageMaker Data Wrangler data flow to remove only values that are outside of those quartiles.
- Use an Amazon SageMaker Data Wrangler bias report to find outliers in the dataset. Use a Data Wrangler data flow to remove outliers based on the bias report.
- Use an Amazon SageMaker Data Wrangler anomaly detection visualization to find outliers in the dataset. Add a transformation to a Data Wrangler data flow to remove outliers. ✓
- Use Amazon Lookout for Equipment to find and remove outliers from the dataset.

Explanation:
The best answer is C because it leverages SageMaker Data Wrangler's built-in anomaly detection visualization feature for identifying outliers and then directly integrates a data transformation step within the same Data Wrangler flow to remove those outliers. This approach minimizes operational overhead by using a visual interface for outlier detection and streamlining the outlier removal process within a single tool.
Here's a breakdown of why the other options are less suitable:

A: Calculating quartiles and manually filtering data is a valid method, but it's more manual and labor-intensive. It involves more coding and might not be as visually intuitive as Data Wrangler's anomaly detection.
B: While bias reports are useful for identifying biases in the dataset, they aren't designed specifically for outlier detection in the same way that the anomaly detection visualization is. They don't automatically pinpoint outliers that skew your dataset.
D: Amazon Lookout for Equipment is designed for predictive maintenance and anomaly detection on industrial equipment data. While it can find outliers, it's more specialized and might be overkill for the initial data cleaning and preparation stage required for training a generic machine-learning model, it might add more complexity than is needed at this stage, and could add to the overhead.
Therefore, the most efficient way to identify and remove outliers with the least operational overhead is by utilizing the Amazon SageMaker Data Wrangler's anomaly detection visualization to find outliers and adding a transformation to remove the outliers.**Supporting Links:**

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Data Wrangler Transformations: https://docs.aws.amazon.com/sagemaker/latest/dg/data-wrangler-transform.html
Amazon Lookout for Equipment: https://aws.amazon.com/lookout-for-equipment/

Question: 307

Exam Heist

A data scientist obtains a tabular dataset that contains 150 correlated features with different ranges to build a regression model. The data scientist needs to achieve more efficient model training by implementing a solution that minimizes impact on the model's performance. The data scientist decides to perform a principal component analysis (PCA) preprocessing step to reduce the number of features to a smaller set of independent features before the data scientist uses the new features in the regression model.

Which preprocessing step will meet these requirements?

- ○ Use the Amazon SageMaker built-in algorithm for PCA on the dataset to transform the data.
- ○ Load the data into Amazon SageMaker Data Wrangler. Scale the data with a Min Max Scaler transformation step. Use the SageMaker built-in algorithm for PCA on the scaled dataset to transform the data. ✓
- ○ Reduce the dimensionality of the dataset by removing the features that have the highest correlation. Load the data into Amazon SageMaker Data Wrangler. Perform a Standard Scaler transformation step to scale the data. Use the SageMaker built-in algorithm for PCA on the scaled dataset to transform the data.
- ○ Reduce the dimensionality of the dataset by removing the features that have the lowest correlation. Load the data into Amazon SageMaker Data Wrangler. Perform a Min Max Scaler transformation step to scale the data. Use the SageMaker built-in algorithm for PCA on the scaled dataset to transform the data.

Explanation:
The most effective preprocessing strategy to meet the specified requirements is option B. Here's why:
PCA is sensitive to the scale of the features. If features have significantly different ranges, PCA will be biased towards features with larger values. This is because PCA aims to find the directions (principal components) that capture the most variance in the data. Features with larger ranges will inherently exhibit more variance and thus unduly influence the principal components.
Therefore, scaling the data is a crucial step before applying PCA. A MinMax Scaler, as used in option B, scales the features to a range between 0 and 1. This ensures that all features contribute equally to the PCA process, regardless of their original ranges.
Amazon SageMaker Data Wrangler provides a user-friendly interface for performing such transformations, making it easy to apply the MinMax Scaler. The SageMaker built-in PCA algorithm then effectively reduces the dimensionality of the scaled data, generating a smaller set of independent features that capture the most important variance in the original dataset.
Option A is insufficient because it applies PCA without scaling, which can lead to biased results. Options C and D, which involve feature selection based on correlation, are not as effective as PCA in creating a set of independent features. While removing highly correlated features can reduce dimensionality, it doesn't guarantee independence among the remaining features and might discard valuable information. Standard Scaler (used in option C) scales data to have zero mean and unit variance. While it works well with normally distributed data, MinMax scaling (used in option B) is often preferred before PCA when feature ranges differ significantly. Options C and D suggest removing features based on correlation *before* PCA, while the intent is to leverage PCA's automatic feature reduction capabilities after scaling. Therefore, option B provides the most complete and effective solution for efficient model training while minimizing the impact on performance.

Amazon SageMaker PCA algorithm: https://docs.aws.amazon.com/sagemaker/latest/dg/pca_HowItWorks.html
Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
MinMax Scaler: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Question: 308

Exam Heist

An online retailer collects the following data on customer orders: demographics, behaviors, location, shipment progress, and delivery time. A data scientist joins all the collected datasets. The result is a single dataset that includes 980 variables.

The data scientist must develop a machine learning (ML) model to identify groups of customers who are likely to respond to a marketing campaign.

Which combination of algorithms should the data scientist use to meet this requirement? (Choose two.)

- ☐ Latent Dirichlet Allocation (LDA)
- ☐ K-means ✓
- ☐ Semantic segmentation

- [x] Principal component analysis (PCA) ✓
- [ ] Factorization machines (FM)

Explanation:
The data scientist needs to perform customer segmentation, grouping customers based on their likelihood to respond to a marketing campaign. Given the high dimensionality (980 variables) of the dataset, dimensionality reduction is a crucial first step.

Principal Component Analysis (PCA) (Option D) is a dimensionality reduction technique that transforms a large number of variables into a smaller set of uncorrelated variables called principal components. These components capture most of the variance in the original data, allowing the model to focus on the most important features and mitigating the curse of dimensionality. Reducing the number of variables simplifies the clustering process and improves model performance.
K-means (Option B) is a clustering algorithm that aims to partition *n* observations into *k* clusters, in which each observation belongs to the cluster with the nearest mean (cluster center or centroid), serving as a prototype of the cluster. After reducing the dimensions using PCA, K-means can effectively group customers based on the principal components. These clusters can represent different customer segments with varying propensities to respond to the marketing campaign.
Latent Dirichlet Allocation (LDA) is generally used for topic modeling in text documents, not for general customer segmentation. Semantic segmentation is an image processing technique, also not applicable here. Factorization Machines (FM) are used for prediction tasks, particularly with sparse data, and are not primarily a clustering or dimensionality reduction method suitable for this problem.
Therefore, PCA is the most appropriate dimensionality reduction technique, and K-means is a well-suited clustering algorithm, making BD the correct answer.
Supporting links:

PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
K-means: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Question: 309

Exam Heist

A machine learning engineer is building a bird classification model. The engineer randomly separates a dataset into a training dataset and a validation dataset. During the training phase, the model achieves very high accuracy. However, the model did not generalize well during validation of the validation dataset. The engineer realizes that the original dataset was imbalanced.

What should the engineer do to improve the validation accuracy of the model?

- ( ) Perform stratified sampling on the original dataset. ✓
- ( ) Acquire additional data about the majority classes in the original dataset.
- ( ) Use a smaller, randomly sampled version of the training dataset.
- ( ) Perform systematic sampling on the original dataset.

Explanation:
The correct answer is **A. Perform stratified sampling on the original dataset.**
Here's a detailed justification:
The problem described points to an imbalanced dataset negatively impacting the model's generalization ability. High training accuracy coupled with poor validation accuracy is a classic sign of overfitting, exacerbated by class imbalance. Because the classes are not represented equally in the training and validation sets, the model learns to predict the majority class well but fails to generalize to the minority class(es), leading to poor validation performance.
Stratified sampling addresses this by ensuring that each class is represented proportionally in both the training and validation datasets. This means the sampling process preserves the percentage of samples for each class as observed in the original dataset. This leads to a more representative validation dataset that accurately reflects the real-world distribution, allowing for a more reliable evaluation of the model's ability to generalize. By maintaining the class proportions, the model will be exposed to more appropriate representations of each class during training, mitigating the bias towards the majority class and improving the model's ability to predict the minority class in the validation set.

Why other options are incorrect:
B. Acquire additional data about the majority classes in the original dataset: Acquiring more data for the majority class will further exacerbate the imbalance, likely making the overfitting worse.

C. Use a smaller, randomly sampled version of the training dataset: Using a smaller, randomly sampled dataset won't necessarily fix the imbalance. It could even make it worse, depending on how the random sampling falls. It also reduces the data the model has to learn from.
D. Perform systematic sampling on the original dataset: Systematic sampling involves selecting samples at regular intervals. This method does not consider the class distribution and may still result in imbalanced training and validation sets. It also doesn't address the root cause of the problem, which is the imbalance in the original dataset.
Therefore, stratified sampling directly addresses the class imbalance issue by ensuring that each class is proportionally represented in both the training and validation sets, leading to a better validation accuracy.
Authoritative Links:

Stratified Sampling: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (While this is about Stratified K-Fold, it illustrates the general principle of stratified sampling)
Imbalanced Datasets: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data (Google's ML Data Prep Guide covers imbalanced data)

Question: 310

Exam Heist

A data engineer wants to perform exploratory data analysis (EDA) on a petabyte of data. The data engineer does not want to manage compute resources and wants to pay only for queries that are run. The data engineer must write the analysis by using Python from a Jupyter notebook.

Which solution will meet these requirements?

- Use Apache Spark from within Amazon Athena.
- Use Apache Spark from within Amazon SageMaker. ✓
- Use Apache Spark from within an Amazon EMR cluster.
- Use Apache Spark through an integration with Amazon Redshift.

Explanation:
Here's a detailed justification for why option B is the best solution, along with explanations for why the other options are less suitable:
Justification for Option B: Use Apache Spark from within Amazon SageMaker
Amazon SageMaker offers a fully managed environment for machine learning, including exploratory data analysis (EDA). SageMaker provides managed notebooks that support Python and Apache Spark via the `pyspark` library. Crucially, SageMaker Studio provides a serverless compute environment, allowing the data engineer to perform EDA without needing to provision or manage EC2 instances directly. This directly addresses the requirement of not managing compute resources.
SageMaker processing jobs can be configured with the required resources to handle the petabyte-scale dataset. SageMaker also supports integration with Amazon S3, where the data is likely stored, enabling efficient data access for Spark. Since the Data Engineer can use Python in a Jupyter notebook and doesn't want to manage resources, Sagemaker is best suited.
Why Other Options Are Less Suitable:

A. Use Apache Spark from within Amazon Athena: While Amazon Athena is serverless and allows querying data directly from S3, Athena's query engine is based on Presto, not Spark. Although Athena can handle large datasets, it's primarily designed for SQL-based queries and not well-suited for running complex Spark-based EDA workflows written in Python from a notebook.
C. Use Apache Spark from within an Amazon EMR cluster: Amazon EMR *is* suitable for processing large datasets with Spark. However, EMR requires the data engineer to provision and manage the EC2 instances that make up the cluster. This violates the requirement of not wanting to manage compute resources. EMR is more cost-effective in long running workloads or if you want full control over the cluster.
D. Use Apache Spark through an integration with Amazon Redshift: Amazon Redshift is a data warehouse designed for analytics. While Redshift can be integrated with Spark, it is better suited for loading and transforming structured data, not necessarily ideal for the interactive and ad-hoc nature of EDA, and still requires resource provisioning.
Supporting Documentation:

Amazon SageMaker: https://aws.amazon.com/sagemaker/
SageMaker Processing Jobs: https://docs.aws.amazon.com/sagemaker/latest/dg/processing-job.html
Amazon Athena: https://aws.amazon.com/athena/
Amazon EMR: https://aws.amazon.com/emr/
Amazon Redshift: https://aws.amazon.com/redshift/

Question: 311

Exam Heist

A data scientist receives a new dataset in .csv format and stores the dataset in Amazon S3. The data scientist will use the dataset to train a machine learning (ML) model.

The data scientist first needs to identify any potential data quality issues in the dataset. The data scientist must identify values that are missing or values that are not valid. The data scientist must also identify the number of outliers in the dataset.

Which solution will meet these requirements with the LEAST operational effort?

- Create an AWS Glue job to transform the data from .csv format to Apache Parquet format. Use an AWS Glue crawler and Amazon Athena with appropriate SQL queries to retrieve the required information.
- Leave the dataset in .csv format. Use an AWS Glue crawler and Amazon Athena with appropriate SQL queries to retrieve the required information.
- Create an AWS Glue job to transform the data from .csv format to Apache Parquet format. Import the data into Amazon SageMaker Data Wrangler. Use the Data Quality and Insights Report to retrieve the required information.
- Leave the dataset in .csv format. Import the data into Amazon SageMaker Data Wrangler. Use the Data Quality and Insights Report to retrieve the required information. ✓

Explanation:

The correct answer is D because it offers the least operational overhead while fulfilling the data quality assessment requirements. Here's a detailed breakdown:

Why D is the best option:

SageMaker Data Wrangler Simplifies Data Quality Assessment: SageMaker Data Wrangler is specifically designed for data preparation and feature engineering for machine learning. Its "Data Quality and Insights Report" feature provides an automated, visual way to identify missing values, invalid values, and outliers. This eliminates the need to manually write complex SQL queries or develop custom code.

Direct .CSV Support: Data Wrangler directly supports importing data from .CSV files stored in S3. This avoids the added step of data transformation, streamlining the workflow and reducing operational effort.

Least Operational Effort: Option D requires only importing the CSV data into Data Wrangler and generating the report. Options that involve AWS Glue and Athena add overhead for setting up crawlers, ETL jobs, and constructing SQL queries, which is a more complex process.

Why other options are less suitable:

A & C (Involving Parquet and Glue Jobs): Converting to Parquet format adds complexity and requires setting up AWS Glue jobs. While Parquet is a great storage format for analytics, it's not necessary for this initial data quality check. The time and resources spent on ETL are not justified when Data Wrangler can directly analyze the .CSV file.

A & B (Involving Athena and SQL): Writing SQL queries to identify missing values, invalid values, and outliers is time-consuming and requires expertise in SQL. SageMaker Data Wrangler provides an automated and visually intuitive alternative to SQL.

Authoritative Links:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/

AWS Glue: https://aws.amazon.com/glue/

Amazon Athena: https://aws.amazon.com/athena/

In summary, option D leverages SageMaker Data Wrangler's specialized features for data quality assessment, offers direct support for .CSV files, and avoids the need for complex ETL or manual SQL querying, thus minimizing operational effort.

Question: 312

Exam Heist

An ecommerce company has developed a XGBoost model in Amazon SageMaker to predict whether a customer will return a purchased item. The dataset is imbalanced. Only 5% of customers return items.

A data scientist must find the hyperparameters to capture as many instances of returned items as possible. The company has a small budget for compute.

How should the data scientist meet these requirements MOST cost-effectively?

- Tune all possible hyperparameters by using automatic model tuning (AMT). Optimize on {"HyperParameterTuningJobObjective": {"MetricName": "validation:accuracy", "Type": "Maximize"}}.
- Tune the csv_weight hyperparameter and the scale_pos_weight hyperparameter by using automatic model tuning (AMT). Optimize on {"HyperParameterTuningJobObjective": {"MetricName": "validation'll", "Type": "Maximize"}}. ✓

- Tune all possible hyperparameters by using automatic model tuning (AMT). Optimize on {"HyperParameterTuningJobObjective": {"MetricName": "validation:f1", "Type": "Maximize"}}.
- Tune the csv_weight hyperparameter and the scale_pos_weight hyperparameter by using automatic model tuning (AMT). Optimize on {"HyperParameterTuningJobObjective": {"MetricName": "validation:f1", "Type": "Minimize"}}.

Explanation:

Here's a detailed justification for why option B is the most suitable solution:

Understanding the Problem:

The core challenge is to optimize an XGBoost model for an imbalanced dataset (only 5% return items) while minimizing computational costs. The goal is to capture as many returned items as possible, which means focusing on recall (correctly identifying positive cases). Accuracy is not the best metric here because a model can achieve high accuracy by simply predicting that almost no one returns items.

Why Option B is the Best Choice:

Targeted Hyperparameter Tuning: Option B focuses on tuning `csv_weight` (also known as `sample_weight` in SageMaker's XGBoost estimator) and `scale_pos_weight`. These hyperparameters are specifically designed to handle imbalanced datasets in XGBoost.

`scale_pos_weight`: Controls the balance of positive and negative weights. Setting it to `sum(negative instances) / sum(positive instances)` is a common starting point for imbalanced datasets.
`csv_weight`: Provides a way to individually weight each sample in the dataset. This can be used if `scale_pos_weight` alone isn't sufficient.

Cost-Effectiveness: Tuning only these two crucial hyperparameters dramatically reduces the search space compared to tuning *all* possible hyperparameters (as in options A and C). A smaller search space translates to fewer training jobs and lower compute costs, which aligns with the small budget constraint.

Appropriate Optimization Metric: The `validation:logloss` metric is not suitable for a case where the goal is to focus on recall. Instead, optimizing "validation:f1" is the best choice in combination with the two hyperparameters. The F1-score is the harmonic mean of precision and recall, providing a good balance. Logloss is more suitable for measuring the model's ability to assign probabilities accurately, not necessarily to maximize recall.

Why Other Options are Less Suitable:

Option A: Tuning *all* hyperparameters is computationally expensive and unnecessary, given that only a few are directly relevant to class imbalance. It violates the budget constraint. It also uses accuracy as the metric which is not suited to imbalanced data.
Option C: Tuning *all* hyperparameters is computationally expensive.
Option D: `validation:f1` should be maximized to get better model.

Key Concepts and Relevant Links:

Imbalanced Datasets: Understanding the challenges and techniques for handling imbalanced data is crucial in machine learning.
XGBoost Hyperparameters: Refer to XGBoost's documentation for details on hyperparameters like `scale_pos_weight`: https://xgboost.readthedocs.io/en/stable/parameter.html
Amazon SageMaker Automatic Model Tuning (AMT): AMT automates the hyperparameter tuning process. https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html
F1-Score: Understanding F1-score is a basic requirement for a data scientist working with classification models.
In summary, option B provides a cost-effective and targeted approach to addressing class imbalance by focusing on the relevant hyperparameters and optimizing for F1. This directly addresses the problem of maximizing the identification of returned items within the given budget constraints.

Question: 313

Exam Heist

A data scientist is trying to improve the accuracy of a neural network classification model. The data scientist wants to run a large hyperparameter tuning job in Amazon SageMaker. However, previous smaller tuning jobs on the same model often ran for several weeks. The ML specialist wants to reduce the computation time required to run the tuning job.

Which actions will MOST reduce the computation time for the hyperparameter tuning job? (Choose two.)

- [x] Use the Hyperband tuning strategy. ✓
- [ ] Increase the number of hyperparameters.
- [x] Set a lower value for the MaxNumberOfTrainingJobs parameter. ✓

- [ ] Use the grid search tuning strategy.
- [ ] Set a lower value for the MaxParallelTrainingJobs parameter.

Explanation:

Here's a detailed justification for why options A and C are the most effective in reducing computation time for a large hyperparameter tuning job in Amazon SageMaker, along with supporting resources:

A. Use the Hyperband tuning strategy.

Hyperband is a resource-allocation-based hyperparameter optimization strategy. It efficiently explores the hyperparameter space by allocating a small budget (e.g., number of epochs, subset of data) to many configurations early on. Poor performing configurations are quickly discarded, and more resources are allocated to the promising ones. This adaptive allocation of resources allows Hyperband to focus on areas of the hyperparameter space that are likely to yield better results. Instead of exhaustively training every configuration, it spends more time training better configurations. This early stopping of poorly performing configurations significantly reduces overall computation time, particularly when dealing with complex models and large datasets. SageMaker supports Hyperband directly.

Reference: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html (Search for Hyperband on this page)

C. Set a lower value for the MaxNumberOfTrainingJobs parameter.

The `MaxNumberOfTrainingJobs` parameter defines the maximum number of training jobs that will be launched during the hyperparameter tuning process. Reducing this number directly limits the total computation time spent on training jobs. While reducing this number might prevent you from exploring the full hyperparameter space, it guarantees a shorter overall tuning time. The best value depends on the trade-off between tuning time and finding optimal hyperparameters, but reducing this value will certainly reduce computation time. Fewer jobs will mean less wall-clock time will be spent.

Reference: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-create-tuning-job.html (Describes the parameters available when creating a tuning job)

Why other options are not as effective:

B. Increase the number of hyperparameters: This will increase the search space and therefore increase the tuning time.

D. Use the grid search tuning strategy: Grid search exhaustively evaluates all possible combinations of hyperparameters within a specified range. This is computationally expensive and time-consuming, especially when dealing with a large number of hyperparameters or a wide range of possible values. It would likely increase, not decrease, the tuning time.

E. Set a lower value for the MaxParallelTrainingJobs parameter: This limits the number of training jobs running concurrently. While it can influence resource utilization and cost, it doesn't necessarily directly reduce the *total* computation time. It can even increase overall time by reducing parallelism. It only affects how fast training jobs execute in parallel, not the total amount of computation the tuning job requires.

Question: 314

Exam Heist

A machine learning (ML) specialist needs to solve a binary classification problem for a marketing dataset. The ML specialist must maximize the Area Under the ROC Curve (AUC) of the algorithm by training an XGBoost algorithm. The ML specialist must find values for the eta, alpha, min_child_weight, and max_depth hyperparameters that will generate the most accurate model.

Which approach will meet these requirements with the LEAST operational overhead?

- ○ Use a bootstrap script to install scikit-learn on an Amazon EMR cluster. Deploy the EMR cluster. Apply k-fold cross-validation methods to the algorithm.
- ○ Deploy Amazon SageMaker prebuilt Docker images that have scikit-learn installed. Apply k-fold cross-validation methods to the algorithm.
- ○ Use Amazon SageMaker automatic model tuning (AMT). Specify a range of values for each hyperparameter. ✓
- ○ Subscribe to an AUC algorithm that is on AWS Marketplace. Specify a range of values for each hyperparameter.

Explanation:

The correct answer is C: Use Amazon SageMaker automatic model tuning (AMT). Specify a range of values for each hyperparameter.

Here's why:

SageMaker Automatic Model Tuning (AMT) is designed for hyperparameter optimization. It automates the process of searching for the best hyperparameter combination for a given machine learning model. This directly addresses the requirement of finding optimal values for eta, alpha, min_child_weight, and max_depth to maximize AUC.

https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html

AMT minimizes operational overhead. It handles the complexities of launching training jobs with different hyperparameter configurations, tracking performance metrics (AUC in this case), and selecting the best-performing model. You only need to define the hyperparameter search space and the objective metric (AUC).
Option A (EMR with scikit-learn and k-fold cross-validation) is less efficient. While it's possible to perform hyperparameter tuning with cross-validation on EMR, it requires significant manual setup and management of the cluster, installation of libraries, and implementation of the tuning loop. This increases operational overhead.
Option B (SageMaker prebuilt images with scikit-learn and k-fold cross-validation) is also less efficient than AMT. It's similar to option A in that it requires manual implementation of the hyperparameter tuning process, even though SageMaker provides the environment.
Option D (AWS Marketplace AUC algorithm) isn't the right approach. While AWS Marketplace offers ML algorithms, it's unlikely to provide a pre-tuned XGBoost model specifically tailored to the marketing dataset and the specified hyperparameters. More importantly, it doesn't allow the ML specialist to explore the hyperparameter space to *maximize* the AUC for the given dataset. Custom model development is crucial for maximizing AUC.
In summary, SageMaker AMT is the most efficient and direct way to optimize hyperparameters for an XGBoost model to maximize AUC, with the least operational overhead. It leverages a managed service designed specifically for this purpose.

Question: 315

Exam Heist

A machine learning (ML) developer for an online retailer recently uploaded a sales dataset into Amazon SageMaker Studio. The ML developer wants to obtain importance scores for each feature of the dataset. The ML developer will use the importance scores to feature engineer the dataset.

Which solution will meet this requirement with the LEAST development effort?

- Use SageMaker Data Wrangler to perform a Gini importance score analysis. ✓
- Use a SageMaker notebook instance to perform principal component analysis (PCA).
- Use a SageMaker notebook instance to perform a singular value decomposition analysis.
- Use the multicollinearity feature to perform a lasso feature selection to perform an importance scores analysis.

Explanation:
The best solution is A: Use SageMaker Data Wrangler to perform a Gini importance score analysis.
Here's why:
SageMaker Data Wrangler is designed to simplify data preparation and feature engineering for ML. It provides a visual interface for performing data transformations without writing code. Crucially, it includes built-in analyses like feature importance, including Gini importance, specifically for understanding feature contributions to model performance. This direct integration requires minimal development effort, as the feature is readily available in the visual workflow.
Option B, Principal Component Analysis (PCA), is a dimensionality reduction technique. While it can identify principal components that explain the variance in the data, it does not directly provide feature importance scores in a way that readily translates to the original features. It requires more steps to interpret.
Option C, Singular Value Decomposition (SVD), is also a dimensionality reduction technique similar to PCA and shares the same drawbacks. It's not a direct method to extract feature importance scores with ease for ML model feature selection and understanding.
Option D, Using multicollinearity feature for lasso feature selection doesn't make sense. Multicollinearity identifies highly correlated features, and Lasso is a regularization technique that can perform feature selection by shrinking coefficients. However, lasso requires writing code and preparing the data, which goes against minimizing development efforts.
Therefore, SageMaker Data Wrangler's built-in Gini importance functionality is the most efficient way to obtain feature importance scores in this scenario.
Amazon SageMaker Data Wrangler DocumentationGini Importance

Question: 316

Exam Heist

A company is setting up a mechanism for data scientists and engineers from different departments to access an Amazon SageMaker Studio domain. Each department has a unique SageMaker Studio domain.

The company wants to build a central proxy application that data scientists and engineers can log in to by using their corporate credentials. The proxy application will authenticate users by using the company's existing Identity provider (IdP). The application will then route users to the appropriate SageMaker Studio domain.

The company plans to maintain a table in Amazon DynamoDB that contains SageMaker domains for each department.

How should the company meet these requirements?

- Use the SageMaker CreatePresignedDomainUrl API to generate a presigned URL for each domain according to the DynamoDB table. Pass the presigned URL to the proxy application. ✓
- Use the SageMaker CreateHumanTaskUi API to generate a UI URL. Pass the URL to the proxy application.
- Use the Amazon SageMaker ListHumanTaskUis API to list all UI URLs. Pass the appropriate URL to the DynamoDB table so that the proxy application can use the URL.
- Use the SageMaker CreatePresignedNotebookInstanceUrl API to generate a presigned URL. Pass the presigned URL to the proxy application.

Explanation:
The correct answer is A. Here's a detailed justification:
The requirement is to provide users from different departments, authenticated via a central proxy with corporate credentials, access to their respective SageMaker Studio domains. The key is to securely provide temporary access URLs based on the user's department, which is mapped to the SageMaker domain in DynamoDB.
Option A leverages the `CreatePresignedDomainUrl` API. This API is specifically designed to generate a presigned URL that grants temporary access to a SageMaker Studio domain. The proxy application can authenticate the user, look up the appropriate domain in DynamoDB, and then use `CreatePresignedDomainUrl` to generate a unique, time-limited URL that directs the user to their SageMaker Studio environment. This ensures that users are routed to the correct domain based on their department without exposing sensitive credentials directly. The presigned URL handles the authorization aspect.
Option B, using `CreateHumanTaskUi`, is incorrect because it's related to creating UIs for human review tasks within SageMaker, not for accessing the SageMaker Studio domain itself.
Option C, using `ListHumanTaskUis`, is also incorrect because it focuses on listing human task UIs and not providing access to the SageMaker Studio domain. Passing these URLs to DynamoDB is unnecessary and irrelevant to the core requirement.
Option D, using `CreatePresignedNotebookInstanceUrl`, is incorrect because it generates presigned URLs for notebook instances, not for the entire SageMaker Studio domain. While notebook instances reside within Studio, this approach doesn't satisfy the requirement of directing users to their specific domain.
Therefore, `CreatePresignedDomainUrl` (Option A) is the only API that directly addresses the need for creating temporary, secure URLs for accessing SageMaker Studio domains based on user identity and domain mapping. It's the most secure and appropriate approach for managing access control in this scenario.
Relevant Documentation:

CreatePresignedDomainUrl API
SageMaker Studio Authentication

Question: 317

Exam Heist

An insurance company is creating an application to automate car insurance claims. A machine learning (ML) specialist used an Amazon SageMaker Object Detection - TensorFlow built-in algorithm to train a model to detect scratches and dents in images of cars. After the model was trained, the ML specialist noticed that the model performed better on the training dataset than on the testing dataset.

Which approach should the ML specialist use to improve the performance of the model on the testing data?

- Increase the value of the momentum hyperparameter.
- Reduce the value of the dropout_rate hyperparameter.
- Reduce the value of the learning_rate hyperparameter
- Increase the value of the L2 hyperparameter. ✓

Explanation:
The problem describes a scenario where a machine learning model (object detection trained with SageMaker) is overfitting, performing well on the training data but poorly on the testing data. This indicates that the model has learned the training data too well, including its noise and specific patterns, and is failing to generalize to unseen data. The goal is to improve the model's ability to generalize and reduce overfitting.
Option D, increasing the L2 regularization hyperparameter, is the most suitable approach. L2 regularization adds a penalty to the loss function based on the squared magnitude of the weights in the model. This encourages the model to use smaller weights, which simplifies the model and prevents it from relying too heavily on specific features in the training data. By reducing the complexity of the model, L2 regularization helps it to generalize better to new data, thus improving performance on the testing dataset. In essence, it discourages the model from learning the noise present in the training data.
Option A, increasing the momentum, might speed up training but doesn't directly address overfitting. Option B, reducing the dropout rate, would actually likely *increase* overfitting, as dropout randomly removes neurons during training, acting as a regularizer. Reducing the learning rate (Option C) might help with convergence, but it primarily affects the speed and stability

of training and is less effective at directly addressing overfitting compared to regularization techniques like L2. Therefore, focusing on controlling model complexity using L2 regularization is the best strategy to improve performance on the testing data.
Further research:

Amazon SageMaker: https://aws.amazon.com/sagemaker/
Regularization (L1, L2): https://developers.google.com/machine-learning/crash-course/regularization/video-lecture
Object Detection with SageMaker: https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection.html

Question: 318

Exam Heist

A developer at a retail company is creating a daily demand forecasting model. The company stores the historical hourly demand data in an Amazon S3 bucket. However, the historical data does not include demand data for some hours.

The developer wants to verify that an autoregressive integrated moving average (ARIMA) approach will be a suitable model for the use case.

How should the developer verify the suitability of an ARIMA approach?

- Use Amazon SageMaker Data Wrangler. Import the data from Amazon S3. Impute hourly missing data. Perform a Seasonal Trend decomposition.
- Use Amazon SageMaker Autopilot. Create a new experiment that specifies the S3 data location. Choose ARIMA as the machine learning (ML) problem. Check the model performance.
- Use Amazon SageMaker Data Wrangler. Import the data from Amazon S3. Resample data by using the aggregate daily total. Perform a Seasonal Trend decomposition. ✓
- Use Amazon SageMaker Autopilot. Create a new experiment that specifies the S3 data location. Impute missing hourly values. Choose ARIMA as the machine learning (ML) problem. Check the model performance.

Explanation:
The correct answer is C. Here's why:
ARIMA models are time series forecasting models that rely on the autocorrelation within the data. To assess ARIMA suitability, we need to analyze the time series characteristics, including seasonality and trend.

Option A is incorrect because imputing hourly data might introduce inaccuracies, affecting the model's performance. Also, ARIMA models don't inherently require data imputation before ST decomposition if missing values are handled differently in preprocessing steps.
Option B is incorrect because SageMaker Autopilot generally automates the process of model selection, training, and tuning. Selecting ARIMA as an option bypasses its primary benefit of exploring different algorithms to discover optimal models. It also recommends imputation directly, which could be suboptimal compared to data aggregation.
Option C is correct because aggregating to daily totals can smooth out some of the variability and potential noise introduced by missing hourly data. Seasonal Trend decomposition is a standard time series analysis technique to check whether a time series is suitable for ARIMA.
Option D is incorrect because using SageMaker Autopilot to directly impute missing hourly values and directly choose ARIMA would restrict Autopilot's exploration of potentially more suitable algorithms. Aggregation before decomposition might be needed before imputation.
Therefore, aggregating to a larger time scale, resampling to obtain the daily total, and then performing decomposition is the proper way to assess the series's structure. This simplifies the data and makes detecting underlying patterns easier. This approach makes a decision about ARIMA's usage.
Supporting Links:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Amazon SageMaker Autopilot: https://aws.amazon.com/sagemaker/autopilot/
Time Series Decomposition: https://otexts.com/fpp3/decomposition.html

Question: 319

Exam Heist

A company decides to use Amazon SageMaker to develop machine learning (ML) models. The company will host SageMaker notebook instances in a VPC. The company stores training data in an Amazon S3 bucket. Company security policy states that SageMaker notebook instances must not have internet connectivity.

Which solution will meet the company's security requirements?

- Connect the SageMaker notebook instances that are in the VPC by using AWS Site-to-Site VPN to encrypt all internet-bound traffic. Configure VPC flow logs. Monitor all network traffic to detect and prevent any malicious activity.
- Configure the VPC that contains the SageMaker notebook instances to use VPC interface endpoints to establish connections for training and hosting. Modify any existing security groups that are associated with the VPC interface endpoint to allow only outbound connections for training and hosting. ✓
- Create an IAM policy that prevents access the internet. Apply the IAM policy to an IAM role. Assign the IAM role to the SageMaker notebook instances in addition to any IAM roles that are already assigned to the instances.
- Create VPC security groups to prevent all incoming and outgoing traffic. Assign the security groups to the SageMaker notebook instances.

Explanation:

Here's a detailed justification for why option B is the correct solution:

The core requirement is to isolate SageMaker notebook instances within a VPC without internet connectivity while allowing them to access training data in S3. VPC interface endpoints provide private connectivity to AWS services, including S3, directly within the VPC. This means traffic to S3 doesn't traverse the public internet, adhering to the security policy.

Option B leverages this by configuring VPC interface endpoints specifically for training and hosting (S3 for data access). Modifying the security groups associated with these endpoints allows fine-grained control, ensuring only outbound traffic related to training and hosting is permitted. This limits the potential attack surface and enforces the no-internet-access policy.

Option A is incorrect because AWS Site-to-Site VPN primarily provides secure connections to on-premises networks, not a substitute for private connectivity to AWS services within a VPC. It doesn't inherently block internet access for the notebook instances. VPC Flow Logs and network monitoring, while helpful for auditing, don't prevent internet access by themselves.

Option C is insufficient. While an IAM policy can restrict actions the instances can perform, it doesn't inherently prevent network traffic from leaving the VPC to the internet if the network configuration allows it. Even with a restrictive IAM role, improperly configured routing tables or security groups could still expose the instances. IAM policies are identity based while security is network based.

Option D is overly restrictive. Blocking all incoming and outgoing traffic prevents the notebook instances from accessing S3 or other essential AWS services necessary for training and hosting models. The instances would become unusable for their intended purpose.

In conclusion, option B provides the most secure and functional solution by utilizing VPC interface endpoints for private connectivity to AWS services, combined with security group rules for granular control, satisfying both the security requirements and operational needs of the company.

Supporting Links:

VPC Endpoints: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
SageMaker Security: https://docs.aws.amazon.com/sagemaker/latest/dg/security.html
Security Groups: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html

Question: 320

Exam Heist

A machine learning (ML) engineer uses Bayesian optimization for a hyperpara meter tuning job in Amazon SageMaker. The ML engineer uses precision as the objective metric.

The ML engineer wants to use recall as the objective metric. The ML engineer also wants to expand the hyperparameter range for a new hyperparameter tuning job. The new hyperparameter range will include the range of the previously performed tuning job.

Which approach will run the new hyperparameter tuning job in the LEAST amount of time?

- Use a warm start hyperparameter tuning job. ✓
- Use a checkpointing hyperparameter tuning job.
- Use the same random seed for the hyperparameter tuning job.
- Use multiple jobs in parallel for the hyperparameter tuning job.

Explanation:

The correct answer is A: Use a warm start hyperparameter tuning job. Here's why:

Warm start tuning leverages knowledge from previous hyperparameter tuning jobs. Because the ML engineer is expanding the hyperparameter range to include the previous range, a warm start can significantly reduce tuning time. SageMaker uses the results of the previous job as a starting point, focusing its search on the new expanded range while still exploiting the optimal regions found in the original tuning job. This allows for a more efficient exploration of the expanded search space, as SageMaker doesn't need to rediscover the previously optimized areas.

Checkpointing, while useful for resuming interrupted jobs, doesn't utilize information from a previous completed tuning job to guide the search process in the same way as a warm start. It only resumes a partially completed job, not inform a new job with a modified search space or objective metric.
Using the same random seed (option C) will only produce the same results if the hyperparameter range and objective metric remained unchanged. Since the objective metric is now recall (instead of precision) and the range has been expanded, using the same seed won't speed up the search process and might even hinder finding the new optimal configuration.
Running multiple jobs in parallel (option D) can reduce the overall elapsed time to get a result, but it is not as efficient as a warm start when the hyperparameter range is expanded and includes the previous range because it does not utilize prior knowledge. Parallel jobs work independently, without any guidance of previously learned patterns.
Therefore, a warm start hyperparameter tuning job offers the most efficient method because it builds upon the knowledge gained from the previous tuning job, allowing for faster convergence to the optimal hyperparameter configuration with the new objective and hyperparameter range.
Relevant link:

SageMaker Hyperparameter Tuning Warm Start

Question: 321

Exam Heist

A news company is developing an article search tool for its editors. The search tool should look for the articles that are most relevant and representative for particular words that are queried among a corpus of historical news documents.

The editors test the first version of the tool and report that the tool seems to look for word matches in general. The editors have to spend additional time to filter the results to look for the articles where the queried words are most important. A group of data scientists must redesign the tool so that it isolates the most frequently used words in a document. The tool also must capture the relevance and importance of words for each document in the corpus.

Which solution meets these requirements?

- Extract the topics from each article by using Latent Dirichlet Allocation (LDA) topic modeling. Create a topic table by assigning the sum of the topic counts as a score for each word in the articles. Configure the tool to retrieve the articles where this topic count score is higher for the queried words.
- Build a term frequency for each word in the articles that is weighted with the article's length. Build an inverse document frequency for each word that is weighted with all articles in the corpus. Define a final highlight score as the product of both of these frequencies. Configure the tool to retrieve the articles where this highlight score is higher for the queried words. ✓
- Download a pretrained word-embedding lookup table. Create a titles-embedding table by averaging the title's word embedding for each article in the corpus. Define a highlight score for each word as inversely proportional to the distance between its embedding and the title embedding. Configure the tool to retrieve the articles where this highlight score is higher for the queried words.
- Build a term frequency score table for each word in each article of the corpus. Assign a score of zero to all stop words. For any other words, assign a score as the word’s frequency in the article. Configure the tool to retrieve the articles where this frequency score is higher for the queried words.

Explanation:
The correct answer is **B**, which leverages Term Frequency-Inverse Document Frequency (TF-IDF). Let's break down why this is the best solution and why the others aren't as suitable:
Why TF-IDF (Option B) is the optimal approach:

Term Frequency (TF): This component measures how often a word appears in a document. By weighting it with the article's length, we normalize the count, preventing longer articles from dominating the results simply because they have more words overall. This addresses the requirement of finding *frequently used* words *within a specific document*.
Inverse Document Frequency (IDF): This component measures how rare a word is across the entire corpus of articles. Common words like "the," "a," or "is" will have a low IDF, while more specialized or unique terms will have a higher IDF. This satisfies the need to capture the *relevance and importance* of words. A word that appears frequently in a document *and* is relatively rare across all documents is likely a key term for that specific document.
Highlight Score (TF * IDF): Multiplying TF and IDF gives a score that reflects both the local frequency (importance within the document) and global rarity (importance across the entire corpus). Articles with high TF-IDF scores for the queried words are precisely those where the words are both frequently used and distinctive, aligning perfectly with the problem requirements.
Directly Addresses the Problem: The initial problem was that the search tool found word matches in general, requiring the editors to filter the results. TF-IDF tackles this by specifically highlighting articles where queried words have high relevance and importance, not just high frequency.
Why the other options are less suitable:

A (LDA): LDA performs topic modeling, grouping words into topics and assigning documents to those topics. While LDA can be useful for understanding the overall themes within the corpus, it doesn't directly address the task of finding the *most*

important words for a *specific document* relative to a given query. It's more about identifying overall topic distributions.
C (Word Embeddings and Title Embeddings): This approach focuses on semantic similarity between words and titles. While useful for suggesting related articles, it doesn't directly address the requirements of identifying the most frequently used and relevant words for a specific document. The distance from a word embedding to the title embedding is more about general relatedness than importance within the article.
D (Term Frequency only with Stop Word Removal): Simply using term frequency, even with stop word removal, doesn't account for the importance of a word across the entire corpus. Common, non-stop words can still dominate the results, leading to the same problem the data scientists are trying to solve. It lacks the crucial IDF component.
Authoritative Links:

TF-IDF: https://en.wikipedia.org/wiki/Tf%E2%80%93idf
Latent Dirichlet Allocation (LDA): https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
Word Embeddings: https://en.wikipedia.org/wiki/Word_embedding
In summary, TF-IDF (Option B) directly addresses the requirement of isolating the most frequently used words in a document while capturing the relevance and importance of those words for each document in the corpus, making it the ideal solution for the news company's article search tool.

Question: 322

Exam Heist

A growing company has a business-critical key performance indicator (KPI) for the uptime of a machine learning (ML) recommendation system. The company is using Amazon SageMaker hosting services to develop a recommendation model in a single Availability Zone within an AWS Region.

A machine learning (ML) specialist must develop a solution to achieve high availability. The solution must have a recovery time objective (RTO) of 5 minutes.

Which solution will meet these requirements with the LEAST effort?

- ○ Deploy multiple instances for each endpoint in a VPC that spans at least two Regions.
- ○ Use the SageMaker auto scaling feature for the hosted recommendation models.
- ○ Deploy multiple instances for each production endpoint in a VPC that spans least two subnets that are in a second Availability Zone. ✓
- ○ Frequently generate backups of the production recommendation model. Deploy the backups in a second Region.

Explanation:
The correct answer is C because it addresses the high availability requirement within the stipulated RTO of 5 minutes using SageMaker's built-in capabilities with minimal operational overhead.
Here's a breakdown:

High Availability and Fault Tolerance: Deploying multiple instances across Availability Zones (AZs) within a single Region inherently provides fault tolerance. If one AZ experiences an outage, the endpoint continues serving traffic using instances in the other AZ. https://aws.amazon.com/reliability/availability-zones/
Recovery Time Objective (RTO): The RTO of 5 minutes is met due to the automatic failover mechanism in place. When an instance in one AZ fails, the load balancer automatically redirects traffic to the healthy instances in the other AZ. This failover occurs quickly, usually within minutes, fulfilling the RTO requirement.
Least Effort: This approach leverages the existing SageMaker hosting infrastructure and requires minimal changes to the model deployment pipeline. No complex cross-region replication or backup/restore procedures are necessary.
Why other options are incorrect:

A (Multiple Regions): While deploying across Regions provides greater disaster recovery capabilities, it adds significant complexity to the deployment and management process. It involves data replication, cross-region load balancing, and potentially longer failover times due to network latency. The complexity is unneeded for an RTO of just 5 minutes.
B (Auto Scaling): Auto scaling helps to dynamically adjust the number of instances based on traffic load, but doesn't guarantee high availability in the event of an Availability Zone outage. It primarily focuses on performance and cost optimization.
D (Backups and Second Region): Relying on backups and deploying to another Region involves significant delays in restoring service. The process of restoring a model from backup to another Region is unlikely to meet the 5-minute RTO. Furthermore, it's more operationally complex.

Question: 323

Exam Heist

A global company receives and processes hundreds of documents daily. The documents are in printed .pdf format or .jpg format.

A machine learning (ML) specialist wants to build an automated document processing workflow to extract text from specific fields from the documents and to classify the documents. The ML specialist wants a solution that requires low maintenance.

Which solution will meet these requirements with the LEAST operational effort?

- Use a PaddleOCR model in Amazon SageMaker to detect and extract the required text and fields. Use a SageMaker text classification model to classify the document.
- Use a PaddleOCR model in Amazon SageMaker to detect and extract the required text and fields. Use Amazon Comprehend to classify the document.
- Use Amazon Textract to detect and extract the required text and fields. Use Amazon Rekognition to classify the document.
- Use Amazon Textract to detect and extract the required text and fields. Use Amazon Comprehend to classify the document. ✓

Explanation:
The correct answer is D because it leverages managed AWS services, minimizing operational overhead. Amazon Textract is designed specifically for optical character recognition (OCR) and extracting text and structured data from documents. It automatically handles the complexities of document processing, including layout analysis and table extraction, reducing the need for custom code or model training. [https://aws.amazon.com/textract/]
Amazon Comprehend is a natural language processing (NLP) service that provides pre-trained models for text classification. Using Comprehend simplifies the classification task as it doesn't require the ML specialist to train and manage a custom classification model in SageMaker, further reducing operational burden. [https://aws.amazon.com/comprehend/]
Option A involves managing a PaddleOCR model within SageMaker, which necessitates handling infrastructure, model deployment, and scaling. Training and maintaining a custom text classification model in SageMaker also adds operational complexity.
Option B still requires managing a PaddleOCR model within SageMaker, incurring operational overhead.
Option C uses Amazon Textract which is good, however, Amazon Rekognition is primarily designed for image analysis and object detection, not document classification based on text content, making Comprehend a more suitable and efficient choice for this task. Using Rekognition for document classification would require custom model development and training, increasing operational efforts.

Question: 324

Exam Heist

A company wants to detect credit card fraud. The company has observed that an average of 2% of credit card transactions are fraudulent. A data scientist trains a classifier on a year's worth of credit card transaction data. The classifier needs to identify the fraudulent transactions. The company wants to accurately capture as many fraudulent transactions as possible.

Which metrics should the data scientist use to optimize the classifier? (Choose two.)

- [ ] Specificity
- [ ] False positive rate
- [ ] Accuracy
- [x] F1 score ✓
- [x] True positive rate ✓

Explanation:
The goal is to maximize the detection of fraudulent transactions in a credit card fraud detection system. Since fraudulent transactions are relatively rare (2%), this is an imbalanced classification problem. Let's analyze the options:

A. Specificity: Specificity measures the proportion of actual negatives (non-fraudulent transactions) that are correctly identified as such. While important, maximizing specificity alone could lead the model to classify most transactions as non-fraudulent, missing many actual fraudulent cases.
B. False Positive Rate: The false positive rate is the proportion of non-fraudulent transactions incorrectly classified as fraudulent. Minimizing this is desirable but not the primary focus when the goal is to catch as many fraudulent transactions as possible.
C. Accuracy: Accuracy can be misleading in imbalanced datasets. A high accuracy could be achieved by simply predicting most transactions as non-fraudulent, since the majority are indeed non-fraudulent. This would result in a poor detection of actual fraud.

D. F1 Score: The F1 score is the harmonic mean of precision and recall. Recall (true positive rate) is crucial in this scenario. F1 score balances precision and recall, offering a good metric to optimize when dealing with imbalanced classes and the need to capture fraudulent transactions effectively.
E. True Positive Rate (Recall): True positive rate, also known as recall, measures the proportion of actual fraudulent transactions that are correctly identified as fraudulent. This aligns directly with the company's objective of accurately capturing as many fraudulent transactions as possible. Maximizing recall ensures that the model identifies a high percentage of the fraudulent activities.
Therefore, the best metrics to optimize are the F1 score, as it balances precision and recall, and true positive rate (recall), because the primary goal is to maximize the identification of actual fraudulent transactions. F1 is a good single metric to optimize, while analyzing TPR (recall) can give a more direct view into the models performance regarding the main business objective. Specificity is less important, accuracy is misleading, and false positive rate is important but not the primary focus.
Relevant links for further research:

Imbalanced Datasets: https://developers.google.com/machine-learning/crash-course/classification/imbalanced-data
Precision and Recall: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
F1 Score: https://en.wikipedia.org/wiki/F-score

Question: 325

Exam Heist

A data scientist is designing a repository that will contain many images of vehicles. The repository must scale automatically in size to store new images every day. The repository must support versioning of the images. The data scientist must implement a solution that maintains multiple immediately accessible copies of the data in different AWS Regions.

Which solution will meet these requirements?

- Amazon S3 with S3 Cross-Region Replication (CRR) ✓
- Amazon Elastic Block Store (Amazon EBS) with snapshots that are shared in a secondary Region
- Amazon Elastic File System (Amazon EFS) Standard storage that is configured with Regional availability
- AWS Storage Gateway Volume Gateway

Explanation:
The correct answer is A: Amazon S3 with S3 Cross-Region Replication (CRR). Here's why:

Scalability: Amazon S3 is designed for virtually unlimited scalability, which is essential for a repository that needs to grow daily.
Versioning: S3 natively supports versioning, allowing you to maintain multiple versions of your image files over time. This is crucial for tracking changes and reverting to previous states.
Cross-Region Replication (CRR): S3 CRR automatically and asynchronously replicates objects across S3 buckets in different AWS Regions. This satisfies the requirement for multiple, immediately accessible copies in different Regions, providing high availability and disaster recovery capabilities.
Let's analyze why the other options are not suitable:

B. Amazon Elastic Block Store (Amazon EBS) with snapshots: EBS is block storage, not ideal for object storage like images. While snapshots can be created and shared across Regions, restoring EBS volumes from snapshots involves manual intervention and is not designed for immediate accessibility of multiple copies.
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-creating-snapshot.html
C. Amazon Elastic File System (Amazon EFS): EFS is a shared file system, which might work, but CRR is not natively supported. While EFS Replication is available, it only supports replicating to one region. EFS is also generally more expensive than S3 for object storage, making S3 a cost-effective option. https://docs.aws.amazon.com/efs/latest/ug/efs-replication.html
D. AWS Storage Gateway Volume Gateway: Volume Gateway provides block storage via a hybrid cloud approach by creating volumes stored in S3. It is not natively designed for managing a large image repository with built-in versioning and immediately available copies in multiple regions. Volume gateway is more suitable for on-premise applications needing block storage backed by AWS. https://docs.aws.amazon.com/storagegateway/latest/userguide/WhatIsStorageGateway.html
In summary, S3 with CRR fulfills all requirements: scalability, versioning, and multiple immediately accessible copies of data in different AWS Regions, making it the most appropriate solution.

Question: 326

Exam Heist

An ecommerce company wants to update a production real-time machine learning (ML) recommendation engine API that uses Amazon SageMaker. The company wants to release a new model but does not want to make changes to applications that rely on the API. The company also wants to evaluate the performance of the new model in production traffic before the company fully rolls out the new model to all users.

Which solution will meet these requirements with the LEAST operational overhead?

- Create a new SageMaker endpoint for the new model. Configure an Application Load Balancer (ALB) to distribute traffic between the old model and the new model.
- Modify the existing endpoint to use SageMaker production variants to distribute traffic between the old model and the new model. ✓
- Modify the existing endpoint to use SageMaker batch transform to distribute traffic between the old model and the new model.
- Create a new SageMaker endpoint for the new model. Configure a Network Load Balancer (NLB) to distribute traffic between the old model and the new model.

Explanation:
Option B, modifying the existing SageMaker endpoint with production variants, is the best solution because it directly addresses the requirements of minimal operational overhead, A/B testing in production, and no API changes for consumers. SageMaker production variants allow deploying multiple versions of a model to the same endpoint, distributing traffic among them according to defined weights. This avoids the need to manage separate endpoints and load balancers, significantly reducing operational complexity.
Option A and D, using new SageMaker endpoints with Application Load Balancer (ALB) or Network Load Balancer (NLB), respectively, introduces unnecessary overhead. While load balancers can distribute traffic, managing two SageMaker endpoints and configuring the load balancer adds complexity compared to using production variants within a single endpoint. Moreover, the application would need to be configured to use the load balancer instead of directly invoking a sagemaker endpoint, creating an additional operational dependency on the load balancer configuration. It is more efficient to allow SageMaker itself to manage routing between the models.
Option C, using SageMaker batch transform, is inappropriate for real-time inference. Batch transform is designed for offline processing of large datasets, not for serving low-latency predictions in response to real-time requests from an API. The ecommerce company wants to update a *real-time* API. Batch transform doesn't allow for real time inferences.
By using SageMaker production variants, the company can easily shift traffic gradually to the new model, monitor its performance using SageMaker's monitoring tools, and rollback if necessary, all within the existing endpoint. This keeps the API contract stable, minimizing disruption. It also simplifies deployment, monitoring, and management compared to using external load balancers and multiple endpoints.
Authoritative Links:

Amazon SageMaker Production Variants: https://docs.aws.amazon.com/sagemaker/latest/dg/endpoint-traffic.html
Amazon SageMaker Endpoints: https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model.html
Amazon SageMaker Batch Transform: https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html

Question: 327

Exam Heist

A machine learning (ML) specialist at a manufacturing company uses Amazon SageMaker DeepAR to forecast input materials and energy requirements for the company. Most of the data in the training dataset is missing values for the target variable. The company stores the training dataset as JSON files.

The ML specialist develop a solution by using Amazon SageMaker DeepAR to account for the missing values in the training dataset.

Which approach will meet these requirements with the LEAST development effort?

- Impute the missing values by using the linear regression method. Use the entire dataset and the imputed values to train the DeepAR model.
- Replace the missing values with not a number (NaN). Use the entire dataset and the encoded missing values to train the DeepAR model. ✓
- Impute the missing values by using a forward fill. Use the entire dataset and the imputed values to train the DeepAR model.
- Impute the missing values by using the mean value. Use the entire dataset and the imputed values to train the DeepAR model.

Explanation:
The correct answer is B: Replace the missing values with not a number (NaN). Use the entire dataset and the encoded missing values to train the DeepAR model.
Here's why:
DeepAR is a probabilistic forecasting model specifically designed to handle time series data, and it has built-in mechanisms for dealing with missing values. Treating missing values explicitly is generally preferred over imputation when the model supports it because imputation can introduce bias, especially in time series where missing data patterns might be informative.

Option B leverages DeepAR's inherent capability to work with NaN values without the need for preprocessing or imputation. DeepAR handles missing data points by using the observed history and future covariates to infer the likely values during training. This method can lead to more accurate forecasts compared to directly imputing the missing values, as the imputation might not reflect the true underlying patterns in the time series data. Moreover, it requires the least development effort since it avoids the need to implement and tune imputation methods like linear regression, forward fill, or mean imputation. These imputation techniques can be complex and potentially introduce their own biases into the data.
Options A, C, and D all involve imputation strategies (linear regression, forward fill, and mean imputation, respectively). While imputation might seem like a straightforward approach, it introduces a potential for bias and requires additional coding effort. In time series, imputing with methods like mean or even linear regression can distort the actual dynamics of the data, especially if the missing values are not missing completely at random. Forward fill, while better than mean in some cases, might still lead to unrealistic data patterns. Using imputed values is generally less accurate than allowing DeepAR to handle the missing data natively. Because DeepAR is specifically built to handle missing values and leverage other data to infer them, it is the best choice.
Therefore, option B is the most efficient and potentially more accurate approach, making it the best answer.
References:

Amazon SageMaker DeepAR: While not explicitly documenting NaN handling, its probabilistic nature and ability to leverage covariates indicate built-in mechanisms for missing value management in time series forecasting. Researching existing DeepAR implementations and examples will usually confirm that it is capable of handling NaNs.

Question: 328

Exam Heist

A law firm handles thousands of contracts every day. Every contract must be signed. Currently, a lawyer manually checks all contracts for signatures.

The law firm is developing a machine learning (ML) solution to automate signature detection for each contract. The ML solution must also provide a confidence score for each contract page.

Which Amazon Textract API action can the law firm use to generate a confidence score for each page of each contract?

- Use the AnalyzeDocument API action. Set the FeatureTypes parameter to SIGNATURES. Return the confidence scores for each page. ✓
- Use the Prediction API call on the documents. Return the signatures and confidence scores for each page.
- Use the StartDocumentAnalysis API action to detect the signatures. Return the confidence scores for each page.
- Use the GetDocumentAnalysis API action to detect the signatures. Return the confidence scores for each page.

Explanation:
The correct answer is **A. Use the AnalyzeDocument API action. Set the FeatureTypes parameter to SIGNATURES. Return the confidence scores for each page.**
Here's a detailed justification:
Amazon Textract is designed for extracting text and structured data from documents. For the specific task of detecting signatures and providing confidence scores, the `AnalyzeDocument` API action is the most suitable choice. This API allows you to specify the type of analysis you want to perform through the `FeatureTypes` parameter. By setting `FeatureTypes` to `SIGNATURES`, you instruct Textract to specifically look for signatures in the document.
The `AnalyzeDocument` API directly returns the detected signatures along with their corresponding confidence scores. The confidence score represents the level of certainty Textract has that it has correctly identified a signature. This aligns perfectly with the law firm's requirement to automate signature detection and obtain a confidence score for each contract page.
Options B, C, and D are less appropriate. Option B mentions a "Prediction API call," which isn't a standard part of Textract's API structure. Options C and D, `StartDocumentAnalysis` and `GetDocumentAnalysis`, are used for asynchronous document processing. While they *could* be used, `AnalyzeDocument` is simpler and more efficient for this specific scenario as it's a synchronous API that returns results immediately. Asynchronous processing with `StartDocumentAnalysis` and `GetDocumentAnalysis` is better suited for larger documents or when immediate results are not required. Since the prompt doesn't mention large contracts or the tolerance for delayed results, synchronous calls are generally better. `AnalyzeDocument` does the whole job with one call instead of two.
In summary, `AnalyzeDocument` with `FeatureTypes` set to `SIGNATURES` provides the most direct and efficient way to detect signatures and retrieve their associated confidence scores, fulfilling the law firm's needs effectively.
For further research, refer to the Amazon Textract documentation:

AnalyzeDocument API
FeatureTypes Parameter

Question: 329

Exam Heist

A company that operates oil platforms uses drones to photograph locations on oil platforms that are difficult for humans to access to search for corrosion.

Experienced engineers review the photos to determine the severity of corrosion. There can be several corroded areas in a single photo. The engineers determine whether the identified corrosion needs to be fixed immediately, scheduled for future maintenance, or requires no action. The corrosion appears in an average of 0.1% of all photos.

A data science team needs to create a solution that automates the process of reviewing the photos and classifying the need for maintenance.

Which combination of steps will meet these requirements? (Choose three.)

- [x] Use an object detection algorithm to train a model to identify corrosion areas of a photo. ✓
- [x] Use Amazon Rekognition with label detection on the photos. ✓
- [ ] Use a k-means clustering algorithm to train a model to classify the severity of corrosion in a photo.
- [ ] Use an XGBoost algorithm to train a model to classify the severity of corrosion in a photo.
- [x] Perform image augmentation on photos that contain corrosion. ✓
- [ ] Perform image augmentation on photos that do not contain corrosion.

Explanation:

The most appropriate solution for automating corrosion detection and severity classification in oil platform drone imagery involves a combination of object detection, data augmentation for positive samples, and potentially a classification algorithm. Here's a breakdown of why A, B, and E are the best choices:

A. Use an object detection algorithm to train a model to identify corrosion areas of a photo: This is crucial because the problem requires identifying *where* the corrosion is within each image. Object detection algorithms (like Faster R-CNN, YOLO, or SSD) excel at this task, identifying and localizing multiple objects (corrosion instances) within a single image.Object Detection

B. Use Amazon Rekognition with label detection on the photos. Amazon Rekognition is a great way to get a good baseline for label detection on images.Rekognition

E. Perform image augmentation on photos that contain corrosion: Since corrosion only appears in 0.1% of the photos, the dataset is heavily imbalanced. This can lead to a model that is biased towards predicting no corrosion. Image augmentation (e.g., rotations, zooms, flips) on the *positive* examples (photos with corrosion) increases the diversity of the training data and helps the model generalize better to unseen corrosion instances.Data Augmentation

Why the other options are less suitable:

C. Use a k-means clustering algorithm to train a model to classify the severity of corrosion in a photo: K-means is an unsupervised clustering algorithm. It's not directly suitable for classification, as it doesn't learn from labeled data (i.e., corrosion severity levels determined by the engineers). It's useful for discovering hidden patterns but not for supervised classification.K-means clustering

D. Use an XGBoost algorithm to train a model to classify the severity of corrosion in a photo: While XGBoost is a powerful classification algorithm, it's less effective when applied directly to the entire image. XGBoost is better used on features extracted *after* the corrosion areas have been located and processed (e.g., size, shape, texture of the corroded area). In this case, it's appropriate after the object detection part.

F. Perform image augmentation on photos that do not contain corrosion: Augmenting the negative examples is less beneficial than augmenting the positive examples in this scenario. Because there are many of these examples, this will add noise to the training set, which makes it less effective.

Question: 330

Exam Heist

A company maintains a 2 TB dataset that contains information about customer behaviors. The company stores the dataset in Amazon S3. The company stores a trained model container in Amazon Elastic Container Registry (Amazon ECR).

A machine learning (ML) specialist needs to score a batch model for the dataset to predict customer behavior. The ML specialist must select a scalable approach to score the model.

Which solution will meet these requirements MOST cost-effectively?

- Score the model by using AWS Batch managed Amazon EC2 Reserved Instances. Create an Amazon EC2 instance store volume and mount it to the Reserved Instances.
- Score the model by using AWS Batch managed Amazon EC2 Spot Instances. Create an Amazon FSx for Lustre volume and mount it to the Spot Instances. ✓
- Score the model by using an Amazon SageMaker notebook on Amazon EC2 Reserved Instances. Create an Amazon EBS volume and mount it to the Reserved Instances.
- Score the model by using Amazon SageMaker notebook on Amazon EC2 Spot Instances. Create an Amazon Elastic File System (Amazon EFS) file system and mount it to the Spot Instances.

Explanation:
The most cost-effective solution for scoring a batch model on a 2 TB dataset stored in S3 to predict customer behavior is option B, using AWS Batch managed Amazon EC2 Spot Instances with an Amazon FSx for Lustre volume. Here's why:

AWS Batch for Scalability: AWS Batch is a fully managed batch processing service that simplifies running batch computing workloads on AWS. It automatically provisions and manages compute resources, allowing for scalable model scoring. https://aws.amazon.com/batch/
Spot Instances for Cost Optimization: Spot Instances offer significant cost savings compared to On-Demand or Reserved Instances. They leverage spare EC2 capacity in the AWS cloud, potentially reducing costs by up to 90%. Batch workloads are typically fault-tolerant, making them suitable for Spot Instances. https://aws.amazon.com/ec2/spot/
Amazon FSx for Lustre for High-Performance Storage: Amazon FSx for Lustre is a high-performance file system optimized for compute-intensive workloads. It provides fast, shared storage for ML model scoring, enabling efficient data access for the EC2 instances running the model. FSx for Lustre can efficiently read data from S3. https://aws.amazon.com/fsx/lustre/
Why other options are less suitable: Option A proposes Reserved Instances, which, while providing guaranteed capacity, are less cost-effective than Spot Instances for fault-tolerant batch processing. Instance store volumes are ephemeral and not designed for persistent data storage. Options C and D utilize SageMaker Notebooks. While notebooks are suitable for development and interactive exploration, they are not ideal for batch scoring production workloads. Also, Amazon EFS (option D) is not optimized for the high-throughput requirements of large-scale model scoring compared to FSx for Lustre. Additionally, EBS (option C) is attached to a single instance, less scalable than FSx.

Question: 331

Exam Heist

A data scientist is implementing a deep learning neural network model for an object detection task on images. The data scientist wants to experiment with a large number of parallel hyperparameter tuning jobs to find hyperparameters that optimize compute time.

The data scientist must ensure that jobs that underperform are stopped. The data scientist must allocate computational resources to well-performing hyperparameter configurations. The data scientist is using the hyperparameter tuning job to tune the stochastic gradient descent (SGD) learning rate, momentum, epoch, and mini-batch size.

Which technique will meet these requirements with LEAST computational time?

- Grid search
- Random search
- Bayesian optimization
- Hyperband ✓

Explanation:
Here's a detailed justification for why Hyperband is the best choice for this scenario:
The core requirement is to efficiently tune hyperparameters for a deep learning model, specifically optimizing compute time while ensuring poor-performing jobs are stopped and resources are allocated to promising configurations.

Grid search exhaustively tests all possible combinations within a defined hyperparameter space. This is computationally expensive and inefficient, especially with a large number of parameters like in this case (learning rate, momentum, epoch, mini-batch size) because every combination is given equal computational time regardless of its performance.
Random search randomly samples hyperparameter combinations. While better than grid search, it doesn't adaptively allocate resources. Poor-performing configurations can still consume significant compute time.
Bayesian optimization builds a probabilistic model of the objective function (in this case, model performance) to guide the search for optimal hyperparameters. Although it explores the hyperparameter space more intelligently than grid and random search, it still sequentially performs hyperparameter configurations without an early stopping criteria for underperforming jobs, and thus, does not guarantee the least computational time.

Hyperband addresses the key requirements efficiently. It employs an adaptive resource allocation strategy. Hyperband starts by evaluating many hyperparameter configurations with a small amount of resources (e.g., few epochs). Poor-performing configurations are quickly identified and stopped, freeing up resources to be allocated to more promising configurations for longer training durations. This iterative process of allocating more resources to better-performing configurations continues until the best hyperparameter set is found. It's particularly effective when the objective function is noisy (which is common in deep learning) and when the computational cost of evaluating a single configuration is high.
Therefore, **Hyperband** is the best choice because it offers the least computational time by aggressively stopping underperforming jobs and adaptively allocating more resources to well-performing configurations during the hyperparameter tuning process, aligning perfectly with the problem statement's requirements.
Authoritative Links:

Hyperband paper: https://arxiv.org/abs/1603.06560
SageMaker Hyperparameter Tuning Strategies: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html

Question: 332

Exam Heist

An agriculture company wants to improve crop yield forecasting for the upcoming season by using crop yields from the last three seasons. The company wants to compare the performance of its new scikit-learn model to the benchmark.

A data scientist needs to package the code into a container that computes both the new model forecast and the benchmark. The data scientist wants AWS to be responsible for the operational maintenance of the container.

Which solution will meet these requirements?

- Package the code as the training script for an Amazon SageMaker scikit-learn container.
- Package the code into a custom-built container. Push the container to Amazon Elastic Container Registry (Amazon ECR).
- Package the code into a custom-built container. Push the container to AWS Fargate.
- Package the code by extending an Amazon SageMaker scikit-learn container. ✓

Explanation:
The correct answer is **D. Package the code by extending an Amazon SageMaker scikit-learn container.**
Here's a detailed justification:

SageMaker's Built-in Containers: Amazon SageMaker provides pre-built, optimized containers for popular machine learning frameworks like scikit-learn. These containers handle much of the operational overhead, such as setting up the environment, installing dependencies, and managing execution.
Extending Containers: Extending a SageMaker container means creating a new container image that inherits from an existing SageMaker container. This allows you to add custom code (in this case, the new model forecast and the benchmark) without having to build everything from scratch. You can inherit all prebuilt library as well.
Operational Maintenance: SageMaker takes responsibility for the operational maintenance of its containers, including patching, scaling, and monitoring. By extending a SageMaker container, the company offloads much of the operational burden to AWS.
Benchmark and Model Comparison: Both the new scikit-learn model and the benchmark can be packaged into the extended container. The container can then run both models and generate performance metrics for comparison.
Why other options are incorrect:

A. Package the code as the training script for an Amazon SageMaker scikit-learn container: While possible, this approach is primarily designed for training, not real-time inference.
B. Package the code into a custom-built container. Push the container to Amazon Elastic Container Registry (Amazon ECR): Requires you to take on the full responsibility of managing infrastructure, security and operational maintenance.
C. Package the code into a custom-built container. Push the container to AWS Fargate: Similar to Option B, this option involves manual creation of image and you have to manage all aspects of its lifecycle.
Authoritative Links:

SageMaker Pre-built Docker Images
Using Docker containers with SageMaker

Question: 333

Exam Heist

A cybersecurity company is collecting on-premises server logs, mobile app logs, and IoT sensor data. The company backs up the ingested data in an Amazon S3 bucket and sends the ingested data to Amazon OpenSearch Service for further analysis. Currently, the company has a custom ingestion pipeline that is running on Amazon EC2 instances. The company needs to implement a new serverless ingestion pipeline that can automatically scale to handle sudden changes in the data flow.

Which solution will meet these requirements MOST cost-effectively?

- Create two Amazon Data Firehose delivery streams to send data to the S3 bucket and OpenSearch Service. Configure the data sources to send data to the delivery streams.
- Create one Amazon Kinesis data stream. Create two Amazon Data Firehose delivery streams to send data to the S3 bucket and OpenSearch Service. Connect the delivery streams to the data stream. Configure the data sources to send data to the data stream.
- Create one Amazon Data Firehose delivery stream to send data to OpenSearch Service. Configure the delivery stream to back up the raw data to the S3 bucket. Configure the data sources to send data to the delivery stream. ✓
- Create one Amazon Kinesis data stream. Create one Amazon Data Firehose delivery stream to send data to OpenSearch Service. Configure the delivery stream to back up the data to the S3 bucket. Connect the delivery stream to the data stream. Configure the data sources to send data to the data stream.

Explanation:

The most cost-effective solution for a serverless ingestion pipeline that scales automatically while backing up data to S3 and sending it to OpenSearch Service is option C. This approach utilizes a single Amazon Data Firehose delivery stream configured to deliver data directly to OpenSearch Service. Firehose's built-in backup feature allows it to simultaneously back up the raw, ingested data to an S3 bucket.

This simplifies the architecture by avoiding the complexity and cost of using Kinesis Data Streams. Kinesis Data Streams, while powerful for complex stream processing, adds unnecessary overhead for a scenario primarily focused on data delivery and backup. Option A involves creating two separate Firehose streams, which increases costs due to redundant processing and data transfer. Option B and D introduce Kinesis Data Streams, adding complexity and cost without a clear justification, as simple delivery and backup are the core requirements. Firehose automatically scales to handle varying data volumes, fulfilling the automatic scaling requirement of the prompt. By using one Firehose stream with S3 backup enabled, the solution minimizes infrastructure and operational overhead, hence making it the most cost-effective option.

The data sources (on-premises servers, mobile apps, IoT sensors) can be configured to send data directly to the Firehose stream, eliminating the need for intermediate processing or buffering layers beyond Firehose's capabilities.

For further reading:

Amazon Data Firehose Documentation: In-depth information on Firehose features, configuration, and best practices.
Amazon S3 Documentation: Details on storage classes, data management, and backup capabilities within S3.
Amazon OpenSearch Service Documentation: Provides information on configuring OpenSearch Service as a destination for Firehose delivery streams.

Question: 334

Exam Heist

A bank has collected customer data for 10 years in CSV format. The bank stores the data in an on-premises server. A data science team wants to use Amazon SageMaker to build and train a machine learning (ML) model to predict churn probability. The team will use the historical data. The data scientists want to perform data transformations quickly and to generate data insights before the team builds a model for production.

Which solution will meet these requirements with the LEAST development effort?

- Upload the data into the SageMaker Data Wrangler console directly. Perform data transformations and generate insights within Data Wrangler.
- Upload the data into an Amazon S3 bucket. Allow SageMaker to access the data that is in the bucket. Import the data from the S3 bucket into SageMaker Data Wrangler. Perform data transformations and generate insights within Data Wrangler. ✓
- Upload the data into the SageMaker Data Wrangler console directly. Allow SageMaker and Amazon QuickSight to access the data that is in an Amazon S3 bucket. Perform data transformations in Data Wrangler and save the transformed data into a second S3 bucket. Use QuickSight to generate data insights.
- Upload the data into an Amazon S3 bucket. Allow SageMaker to access the data that is in the bucket. Import the data from the bucket into SageMaker Data Wrangler. Perform data transformations in Data Wrangler. Save the data into a second S3 bucket. Use a SageMaker Studio notebook to generate data insights.

Explanation:

The correct answer is B. Here's why:

Option B is the most efficient and direct approach for the scenario. It leverages the strengths of Amazon S3 and SageMaker Data Wrangler for data preparation and exploration.

Data Storage in S3: Storing the historical data in an Amazon S3 bucket is a standard practice for large datasets in AWS. S3 provides scalable, durable, and cost-effective storage. SageMaker can directly access data stored in S3, making it a seamless integration.
Data Wrangler for Transformation: SageMaker Data Wrangler is designed for interactive data exploration, cleaning, and feature engineering. It offers a visual interface for applying various transformations without writing code, fulfilling the requirement for quick data transformations.
Data Wrangler for Insights: Data Wrangler also provides built-in visualizations and statistical analysis tools to generate data insights directly within the tool. This eliminates the need for separate services for initial data exploration.
Least Development Effort: Option B requires minimal development effort. The data scientists can focus on using Data Wrangler's built-in capabilities rather than writing code for data access, transformation, and visualization.

Why other options are not the best:

A: Data Wrangler is not designed for direct uploads of large CSV files directly from a local machine to the console. It works best when data is accessed from storage like S3.
C: While QuickSight can provide data insights, it adds unnecessary complexity. Data Wrangler already provides insight generation features. Introducing a separate service increases the effort. Additionally, Data Wrangler does not necessitate direct access to the initial raw data in S3 for QuickSight.
D: Using a SageMaker Studio notebook to generate data insights after transformation increases the development effort. Data Wrangler's built-in features provide the desired insights directly. Saving to a second S3 bucket before generating insights is also an unneeded step.
In summary, option B aligns best with the requirements for quick data transformations, insight generation, and minimal development effort by leveraging S3 for storage and Data Wrangler for both transformation and initial analysis.
Supporting Documentation:

Amazon S3: https://aws.amazon.com/s3/
SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/

Question: 335

Exam Heist

A media company wants to deploy a machine learning (ML) model that uses Amazon SageMaker to recommend new articles to the company's readers. The company's readers are primarily located in a single city.

The company notices that the heaviest reader traffic predictably occurs early in the morning, after lunch, and again after work hours. There is very little traffic at other times of day. The media company needs to minimize the time required to deliver recommendations to its readers. The expected amount of data that the API call will return for inference is less than 4 MB.

Which solution will meet these requirements in the MOST cost-effective way?

- ◯ Real-time inference with auto scaling ✓
- ◯ Serverless inference with provisioned concurrency
- ◯ Asynchronous inference
- ◯ A batch transform task

Explanation:
The correct answer is A: **Real-time inference with auto scaling**. Here's why:

Real-time Inference: This is crucial because the company needs to deliver recommendations quickly (minimize the time required). Real-time inference provides immediate responses to requests, which is necessary for a recommendation system that needs to provide suggestions as users interact with the platform.
Auto Scaling: The reader traffic is highly variable, with predictable peaks and troughs. Auto scaling allows the SageMaker endpoint to automatically adjust the number of instances based on the incoming request volume. During peak hours, more instances are spun up to handle the increased load, ensuring low latency. During off-peak hours, instances are scaled down, reducing costs.
Cost-Effectiveness: While serverless inference with provisioned concurrency (option B) might seem attractive, it's generally more cost-effective for infrequent or unpredictable workloads. In this scenario, the traffic patterns are predictable. Auto scaling for real-time inference is likely cheaper because you only pay for the instances needed to handle the current load, scaling down when traffic is low. Serverless with provisioned concurrency might keep a certain level of concurrency always active, leading to unnecessary costs during off-peak hours.
Asynchronous Inference: (Option C) Asynchronous inference is not suitable because it's designed for large payloads or latency-tolerant applications. It queues requests and processes them later, which is unacceptable when the requirement is to minimize the time required to deliver recommendations.

Batch Transform: (Option D) Batch transform is for processing large datasets offline. It's not suitable for real-time recommendations.
Therefore, real-time inference with auto scaling provides the best balance of low latency and cost-effectiveness for this specific scenario.*Authoritative Links:*

Amazon SageMaker Real-time Inference: https://aws.amazon.com/sagemaker/realtime-inference/
Amazon SageMaker Auto Scaling: https://docs.aws.amazon.com/sagemaker/latest/dg/endpoint-auto-scaling.html

Question: 336

Exam Heist

A machine learning (ML) engineer is using Amazon SageMaker automatic model tuning (AMT) to optimize a model's hyperparameters. The ML engineer notices that the tuning jobs take a long time to run. The tuning jobs continue even when the jobs are not significantly improving against the objective metric.

The ML engineer needs the training jobs to optimize the hyperparameters more quickly.

How should the ML engineer configure the SageMaker AMT data types to meet these requirements?

- Set Strategy to the Bayesian value.
- Set RetryStrategy to a value of 1.
- Set ParameterRanges to the narrow range Inferred from previous hyperparameter jobs.
- Set TrainingJobEarlyStoppingType to the AUTO value. ✓

Explanation:
The correct answer is **D. Set TrainingJobEarlyStoppingType to the AUTO value.** Here's why:
The problem statement explicitly states the need to reduce the tuning job duration and stop jobs that aren't significantly improving. SageMaker's Automatic Model Tuning (AMT) offers mechanisms to optimize this process.

Option D directly addresses the problem by enabling automatic early stopping. `TrainingJobEarlyStoppingType = AUTO` allows SageMaker to automatically stop training jobs that are unlikely to yield significant improvement in the objective metric. This prevents resources from being wasted on unproductive jobs, thereby shortening the overall tuning job duration. This is an intelligent stopping policy that dynamically determines when a training job has converged or is unlikely to improve further, based on the training progress.
Option A (Strategy = Bayesian) focuses on the search strategy used to explore the hyperparameter space. While Bayesian optimization is generally efficient, it doesn't inherently stop unproductive jobs. It prioritizes more promising regions of the hyperparameter space, but a poorly performing job within a promising region could still run to completion.
Option B (RetryStrategy = 1) is irrelevant in this scenario. The retry strategy determines what happens if a training job fails due to an infrastructure issue. Retrying a failed job doesn't help in stopping unproductive jobs.
Option C (ParameterRanges = narrow range) can help speed up tuning *if* the initial ranges are too wide. However, reducing the search space too aggressively might prevent finding better solutions, and it doesn't solve the core issue of running unproductive jobs to completion. Early stopping provides a dynamic way to avoid such situation.
Therefore, enabling automatic early stopping is the most effective way to reduce tuning job duration by preventing resource waste on training jobs that are not improving.
Relevant documentation:

Configure hyperparameter tuning resources - Amazon SageMaker
HyperparameterTuningJobConfig - Amazon SageMaker (specifically, the `TrainingJobEarlyStoppingType` field)

Question: 337

Exam Heist

A global bank requires a solution to predict whether customers will leave the bank and choose another bank. The bank is using a dataset to train a model to predict customer loss. The training dataset has 1,000 rows. The training dataset includes 100 instances of customers who left the bank.

A machine learning (ML) specialist is using Amazon SageMaker Data Wrangler to train a churn prediction model by using a SageMaker training job. After training, the ML specialist notices that the model returns only false results. The ML specialist must correct the model so that it returns more accurate predictions.

Which solution will meet these requirements?

- Apply anomaly detection to remove outliers from the training dataset before training.
- Apply Synthetic Minority Oversampling Technique (SMOTE) to the training dataset before training. ✓

- Apply normalization to the features of the training dataset before training.
- Apply undersampling to the training dataset before training.

Explanation:
The problem describes a scenario where a churn prediction model trained on a highly imbalanced dataset (100 churned customers out of 1000 total) consistently predicts negative outcomes (no churn). This indicates that the model is biased towards the majority class (customers who didn't churn) and effectively ignores the minority class (churned customers). The goal is to rebalance the dataset to improve the model's ability to correctly identify churned customers.
Option A (anomaly detection) is incorrect because while outliers can affect model performance, they don't directly address the core issue of class imbalance. Outlier removal might even worsen the problem by potentially removing some of the already limited instances of the minority class.
Option B (SMOTE) is the correct approach. SMOTE (Synthetic Minority Oversampling Technique) is an oversampling technique specifically designed to address class imbalance. It works by creating synthetic samples of the minority class by interpolating between existing minority class instances. This increases the representation of the minority class without simply duplicating existing samples, thus reducing the bias towards the majority class. By generating synthetic churned customer data, SMOTE helps the model learn the characteristics of this group more effectively, improving its ability to predict churn accurately. This directly addresses the issue of the model primarily predicting non-churn due to lack of exposure to churned customer data.
Option C (normalization) is incorrect because normalization is a feature scaling technique that brings all features into a similar range. While normalization is generally a good practice for improving model training, it does not address the fundamental problem of class imbalance. A normalized imbalanced dataset is still an imbalanced dataset.
Option D (undersampling) involves reducing the size of the majority class. In this case, undersampling would mean removing instances of customers who *didn't* churn. Given that the dataset already has a limited number of churned customer instances (only 100), further reducing the dataset size, even for the majority class, can lead to information loss and potentially worse model performance. Undersampling, in general, is usually avoided when the minority class is significantly smaller than the majority class, which is the case here.Therefore, SMOTE is the most appropriate solution to address the class imbalance and improve the churn prediction model's accuracy.
Refer to the following resources for more information:

SMOTE: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
Handling Imbalanced Datasets: https://developers.google.com/machine-learning/data-prep/transform/imbalanced-data

Question: 338

Exam Heist

A banking company provides financial products to customers around the world. A machine learning (ML) specialist collected transaction data from internal customers. The ML specialist split the dataset into training, testing, and validation datasets. The ML specialist analyzed the training dataset by using Amazon SageMaker Clarify. The analysis found that the training dataset contained fewer examples of customers in the 40 to 55 year-old age group compared to the other age groups.

Which type of pretraining bias did the ML specialist observe in the training dataset?

- Difference in proportions of labels (DPL)
- Class imbalance (CI) ✓
- Conditional demographic disparity (CDD)
- Kolmogorov-Smirnov (KS)

Explanation:
The ML specialist observed **Class Imbalance (CI)** in the training dataset. Here's why:

Class Imbalance (CI) occurs when the number of examples for different classes (in this case, age groups) within a dataset is significantly different. The problem description explicitly states that the training dataset contained fewer examples of customers in the 40 to 55 year-old age group compared to other age groups. This discrepancy in representation directly indicates class imbalance.
Difference in Proportions of Labels (DPL) focuses on imbalances in the *outcomes* or *labels* associated with different groups. The question only provides information about the *features* (age groups) and their distribution.
Conditional Demographic Disparity (CDD) requires analysis of the *outcome* given a specific *feature*. CDD detects bias if the model performs better or worse for certain demographic groups *within* the context of specific features. The question does not provide information about the model's performance concerning these demographics.
Kolmogorov-Smirnov (KS) test is a statistical test used to determine if two samples are drawn from the same distribution. While the KS test can detect differences in feature distributions, it's not specifically designed for identifying the *type* of pretraining bias as class imbalance.

Amazon SageMaker Clarify can detect various biases. One common use case is identifying class imbalances, enabling preemptive adjustments to the model training process to mitigate potential unfairness or performance degradation. Addressing class imbalance is a crucial step in ensuring that machine learning models generalize well across all demographic groups, leading to more robust and equitable financial product offerings.
Supporting Links:

Amazon SageMaker Clarify: https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-bias-metrics.html
Class Imbalance: https://developers.google.com/machine-learning/data-validation/identify-skew

Question: 339

Exam Heist

A tourism company uses a machine learning (ML) model to make recommendations to customers. The company uses an Amazon SageMaker environment and set hyperparameter tuning completion criteria to MaxNumberOfTrainingJobs.

An ML specialist wants to change the hyperparameter tuning completion criteria. The ML specialist wants to stop tuning immediately after an internal algorithm determines that tuning job is unlikely to improve more than 1% over the objective metric from the best training job.

Which completion criteria will meet this requirement?

- MaxRuntimeInSeconds
- TargetObjectiveMetricValue
- CompleteOnConvergence ✓
- MaxNumberOfTrainingJobsNotImproving

Explanation:
The correct answer is **C. CompleteOnConvergence**.
Here's a detailed justification:
The core requirement is to stop the hyperparameter tuning job *automatically* when further improvements are unlikely. We need a completion criterion that isn't just a fixed limit (like time or job count) but is sensitive to the *learning process* itself.

CompleteOnConvergence is specifically designed to address this need. It monitors the objective metric (the metric you're trying to optimize, like accuracy or F1-score) and stops the tuning job when it detects that further training jobs are unlikely to yield significant improvements (in this case, more than 1%). This directly aligns with the requirement that the tuning job should halt when the improvement is expected to be less than 1% compared to the best training job seen so far. The underlying algorithm analyzes the trend of the objective metric and, based on a convergence threshold, determines when to stop.
MaxRuntimeInSeconds sets a maximum time limit. It doesn't consider the model's learning progress or whether it's still improving. It just stops after the specified duration, regardless of whether the model has reached a satisfactory performance level or is still making significant progress. This doesn't align with stopping based on algorithm convergence.
TargetObjectiveMetricValue stops the tuning job when a *specific* target value for the objective metric is reached. While this seems related, it's not the right choice here. The requirement is to stop when improvement is unlikely *relative to the best result so far*, not when a predefined absolute target is met. The desired percentage increase relates to the trend of objective metric value itself.
MaxNumberOfTrainingJobsNotImproving is incorrect because it focuses on the number of training jobs that *don't* improve. The tuning job will terminate when the defined number of training jobs don't bring any improvements.
In summary, `CompleteOnConvergence` provides the dynamic stopping behavior desired, monitoring for diminishing returns and halting the tuning job when improvements are deemed unlikely, thereby satisfying the requirement of stopping when improvements are expected to be less than 1% better than the best job so far.
Authoritative links for further research:

AWS SageMaker Hyperparameter Tuning: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works.html
HyperparameterTuningJobConfig:
https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_HyperParameterTuningJobConfig.html

Question: 340

Exam Heist

A car company has dealership locations in multiple cities. The company uses a machine learning (ML) recommendation system to market cars to its customers.

An ML engineer trained the ML recommendation model on a dataset that includes multiple attributes about each car. The dataset includes attributes

such as car brand, car type, fuel efficiency, and price.

The ML engineer uses Amazon SageMaker Data Wrangler to analyze and visualize data. The ML engineer needs to identify the distribution of car prices for a specific type of car.

Which type of visualization should the ML engineer use to meet these requirements?

- Use the SageMaker Data Wrangler scatter plot visualization to inspect the relationship between the car price and type of car.
- Use the SageMaker Data Wrangler quick model visualization to quickly evaluate the data and produce importance scores for the car price and type of car.
- Use the SageMaker Data Wrangler anomaly detection visualization to Identify outliers for the specific features.
- Use the SageMaker Data Wrangler histogram visualization to inspect the range of values for the specific feature. ✓

Explanation:

The question asks how to visualize the distribution of car prices for a specific car type using SageMaker Data Wrangler. The correct answer is D, using a histogram.

A histogram is a graphical representation of the distribution of numerical data. It groups data into bins and displays the number of data points that fall into each bin. This makes it ideal for understanding the range and frequency of car prices for a given car type. By filtering the dataset in Data Wrangler to only include the specific car type, the histogram will then visualize the distribution of prices for only those cars.

Option A, a scatter plot, is useful for visualizing the relationship between two numerical variables. While it could show the relationship between car type and price, it isn't the best tool for understanding the *distribution* of prices for *one* specific car type. It would be cluttered with data points and less clear than a histogram.

Option B, a quick model, produces feature importance scores. This is useful for understanding which features most influence the model's prediction but doesn't directly visualize the price distribution. Although potentially interesting, this addresses a different use case than needed.

Option C, anomaly detection, focuses on identifying outliers, which, again, is a different goal than visualizing the overall price distribution. While helpful for data cleaning, it does not show the price distribution for specific types of cars.

Therefore, a histogram provides the most direct and effective way to visualize the range and distribution of car prices for a specific car type within SageMaker Data Wrangler, directly addressing the engineer's needs. Histograms are the standard visualization technique for this specific scenario.

For more information on histograms and their use in data analysis, consult resources like:

"Histograms: Understanding the Visual Language of Data" by Geckoboard: https://www.geckoboard.com/blog/histogram/
"Histograms" by Math is Fun: https://www.mathsisfun.com/data/histograms.html

Question: 341

Exam Heist

A media company is building a computer vision model to analyze images that are on social media. The model consists of CNNs that the company trained by using images that the company stores in Amazon S3. The company used an Amazon SageMaker training job in File mode with a single Amazon EC2 On-Demand Instance.

Every day, the company updates the model by using about 10,000 images that the company has collected in the last 24 hours. The company configures training with only one epoch. The company wants to speed up training and lower costs without the need to make any code changes.

Which solution will meet these requirements?

- Instead of File mode, configure the SageMaker training job to use Pipe mode. Ingest the data from a pipe.
- Instead of File mode, configure the SageMaker training job to use FastFile mode with no other changes. ✓
- Instead of On-Demand Instances, configure the SageMaker training job to use Spot Instances. Make no other changes,
- Instead of On-Demand Instances, configure the SageMaker training job to use Spot Instances, implement model checkpoints.

Explanation:

The goal is to speed up training and lower costs for a daily SageMaker training job without code changes.

Option B is correct: FastFile mode is designed to improve the speed of data loading compared to File mode. It leverages optimized data transfer mechanisms and caching, potentially reducing the time the training job spends waiting for data. This directly addresses the requirement of speeding up training without code changes.

Option A is incorrect: While Pipe mode is more efficient than File mode, switching to Pipe mode requires code changes to ingest data from a pipe instead of directly from files. This violates the "no code changes" requirement.

Option C is incorrect: Using Spot Instances lowers costs, but they can be interrupted. If the training only runs one epoch and completes quickly, a Spot Instance is potentially a good cost saving method. However if the process is long, it is more likely to be interrupted
Option D is incorrect: Spot Instances with model checkpoints is a good method if you are using Spot Instances to save cost, and have longer training jobs. The use of Spot Instances can be interrupted, therefore model checkpoints are used to recover from interruption and save cost. However this doesn't reduce training time.
Therefore, FastFile mode is the best option.

Question: 342

Exam Heist

A telecommunications company has deployed a machine learning model using Amazon SageMaker. The model identifies customers who are likely to cancel their contract when calling customer service. These customers are then directed to a specialist service team. The model has been trained on historical data from multiple years relating to customer contracts and customer service interactions in a single geographic region.

The company is planning to launch a new global product that will use this model. Management is concerned that the model might incorrectly direct a large number of calls from customers in regions without historical data to the specialist service team.

Which approach would MOST effectively address this issue?

- Enable Amazon SageMaker Model Monitor data capture on the model endpoint. Create a monitoring baseline on the training dataset. Schedule monitoring jobs. Use Amazon CloudWatch to alert the data scientists when the numerical distance of regional customer data fails the baseline drift check. Reevaluate the training set with the larger data source and retrain the model. ✓
- Enable Amazon SageMaker Debugger on the model endpoint. Create a custom rule to measure the variance from the baseline training dataset. Use Amazon CloudWatch to alert the data scientists when the rule is invoked. Reevaluate the training set with the larger data source and retrain the model.
- Capture all customer calls routed to the specialist service team in Amazon S3. Schedule a monitoring job to capture all the true positives and true negatives, correlate them to the training dataset, and calculate the accuracy. Use Amazon CloudWatch to alert the data scientists when the accuracy decreases. Reevaluate the training set with the additional data from the specialist service team and retrain the model.
- Enable Amazon CloudWatch on the model endpoint. Capture metrics using Amazon CloudWatch Logs and send them to Amazon S3. Analyze the monitored results against the training data baseline. When the variance from the baseline exceeds the regional customer variance, reevaluate the training set and retrain the model.

Explanation:
The most effective approach to address the potential issue of the model misdirecting calls from new geographic regions is option A, leveraging SageMaker Model Monitor. Here's why:
Data Drift Detection: The core concern is that the model, trained on a single region's data, may not generalize well to new regions. This is a classic case of data drift – a change in the input data distribution compared to the training data.
SageMaker Model Monitor: SageMaker Model Monitor is specifically designed to detect and alert on data drift. By enabling data capture on the endpoint, it records the input data being sent to the model.
[https://docs.aws.amazon.com/sagemaker/latest/dg/model-monitor.html]
Baseline Creation: Creating a monitoring baseline on the training dataset provides a reference point for the expected data distribution. This baseline represents the characteristics of the data the model was trained on.
Scheduled Monitoring Jobs: Scheduling monitoring jobs periodically compares the captured input data from the new regions to the baseline.
Drift Detection and Alerting: The monitoring jobs calculate the distance between the new region data and the baseline. If the distance exceeds a predefined threshold (indicating significant drift), CloudWatch alerts data scientists.
Retraining: The alert triggers a reevaluation of the training set, incorporating data from the new regions. This allows the model to learn the characteristics of the new regions and improve its generalization ability.
Why other options are less suitable:

B (SageMaker Debugger): SageMaker Debugger primarily focuses on debugging training jobs, identifying issues like vanishing gradients. It's not designed for continuous monitoring of data drift on deployed endpoints.
[https://docs.aws.amazon.com/sagemaker/latest/dg/debugger.html]
C (Capture True Positives/Negatives): While collecting true positives/negatives helps evaluate model performance, it doesn't proactively identify *when* the model is likely to perform poorly due to data drift. It's a reactive approach, detecting issues *after* they occur. Also, it is difficult to capture true positives and negatives easily.
D (CloudWatch Metrics): CloudWatch can monitor endpoint metrics (e.g., CPU utilization, latency), but it doesn't directly analyze the *content* of the input data to detect drift. You could potentially derive indirect metrics related to model outputs, but this is less precise and requires more manual effort than using Model Monitor.

In summary, SageMaker Model Monitor with data capture, baseline creation, scheduled monitoring jobs, and CloudWatch alerts provides a proactive and automated solution for detecting and addressing data drift, ensuring the model's accuracy and effectiveness in new geographic regions.

Question: 343

Exam Heist

A machine learning (ML) engineer is creating a binary classification model. The ML engineer will use the model in a highly sensitive environment.

There is no cost associated with missing a positive label. However, the cost of making a false positive inference is extremely high.

What is the most important metric to optimize the model for in this scenario?

- Accuracy
- Precision ✓
- Recall
- F1

Explanation:

The correct answer is **B. Precision**.
Here's a detailed justification:
In the context of a binary classification model where the cost of a false positive is extremely high, the primary goal is to minimize these incorrect positive predictions. Precision directly addresses this concern. Precision measures the proportion of correctly predicted positive cases out of all instances predicted as positive. A high precision score indicates that when the model predicts a positive outcome, it is highly likely to be correct, thus minimizing the number of false positives.
Accuracy, while a general measure of overall correctness, doesn't prioritize the reduction of false positives specifically. Recall focuses on minimizing false negatives, which is not a primary concern in this scenario as the cost of missing a positive label is negligible. F1-score is the harmonic mean of precision and recall, and while useful in many classification problems, it doesn't solely emphasize the minimization of false positives when they are critically costly.
Therefore, optimizing for precision ensures that the model makes positive predictions only when it is highly confident, thus avoiding the costly consequences of false positives. In situations where false positives are detrimental (e.g., medical diagnoses, fraud detection), maximizing precision is paramount, even if it means sacrificing some recall. High precision in this scenario directly translates to fewer costly errors.
For further reading:

Precision and Recall: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall
Evaluating Machine Learning Models: https://aws.amazon.com/blogs/machine-learning/evaluating-machine-learning-models/ (AWS blog on model evaluation)

Question: 344

Exam Heist

An ecommerce company discovers that the search tool for the company's website is not presenting the top search results to customers. The company needs to resolve the issue so the search tool will present results that customers are most likely to want to purchase.

Which solution will meet this requirement with the LEAST operational effort?

- Use the Amazon SageMaker BlazingText algorithm to add context to search results through query expansion.
- Use the Amazon SageMaker XGBoost algorithm to improve candidate ranking.
- Use Amazon CloudSearch and sort results by the search relevance score. ✓
- Use Amazon CloudSearch and sort results by the geographic location.

Explanation:

The correct answer is C, using Amazon CloudSearch and sorting results by search relevance score. This approach requires the least operational effort while addressing the core problem of improving search result ranking based on customer purchase likelihood.
Here's why:

Amazon CloudSearch: It is a managed service specifically designed for building search applications. It handles the complexities of indexing, scaling, and maintaining a search engine.

Search Relevance Score: CloudSearch automatically calculates a relevance score for each document based on the search query. This score reflects how well the document matches the query terms and other factors defined during configuration.
Sorting by Relevance: By sorting results by the search relevance score, the solution ensures that the most relevant items (those most likely to be purchased) appear at the top of the search results.
Minimal Effort: Compared to other options, CloudSearch requires less custom development and machine learning expertise. The relevance scoring is built-in, minimizing the need for extensive model training and deployment.
Option A (SageMaker BlazingText for query expansion) can improve search recall but doesn't directly address the ranking issue or customer purchase likelihood. Query expansion can introduce irrelevant results. Option B (SageMaker XGBoost for candidate ranking) would require significantly more effort in data preparation, feature engineering, model training, and deployment. It's overkill for the given problem and increases operational complexity. Option D (CloudSearch by geographic location) is irrelevant to the goal of presenting the most purchasable items. Location is not related to purchase likelihood. Therefore, using CloudSearch and its built-in relevance ranking provides the most straightforward and least operationally intensive solution to improve search results based on customer purchase intent.
Supporting Documentation:

Amazon CloudSearch: https://aws.amazon.com/cloudsearch/
Amazon CloudSearch Relevance Ranking: https://docs.aws.amazon.com/cloudsearch/latest/developerguide/configuring-ranking.html

Question: 345

Exam Heist

A machine learning (ML) specialist collected daily product usage data for a group of customers. The ML specialist appended customer metadata such as age and gender from an external data source.

The ML specialist wants to understand product usage patterns for each day of the week for customers in specific age groups. The ML specialist creates two categorical features named dayofweek and binned_age, respectively.

Which approach should the ML specialist use discover the relationship between the two new categorical features?

- Create a scatterplot for day_of_week and binned_age.
- Create crosstabs for day_of_week and binned_age. ✓
- Create word clouds for day_of_week and binned_age.
- Create a boxplot for day_of_week and binned_age.

Explanation:
The task is to discover the relationship between two categorical features: `dayofweek` and `binned_age`. Let's analyze why the correct answer is creating crosstabs.

Crosstabs (Contingency Tables): Crosstabs, also known as contingency tables, are ideal for summarizing and analyzing the relationship between two or more categorical variables. They display the frequency distribution of the variables, showing how many observations fall into each combination of categories. This allows for a direct comparison of the counts or proportions within each combination, revealing any associations or dependencies between the categorical features. In this case, a crosstab would show the count of customers in each age bin for each day of the week, highlighting usage patterns across different age groups on different days.
Scatterplots: Scatterplots are used to visualize the relationship between two continuous numerical variables. Because `dayofweek` and `binned_age` are categorical, a scatterplot would not provide meaningful insights. The points would simply be scattered randomly across the limited set of categorical values, without revealing any relationship.
Word Clouds: Word clouds are used to visualize the frequency of words in a text corpus. They are not suitable for analyzing categorical data unless the categories themselves are textual.
Boxplots: Boxplots are used to visualize the distribution of a continuous variable for different categories of a categorical variable. While `dayofweek` could be used as the categorical variable, `binned_age` is also categorical. A boxplot requires a numerical value as the main variable being analyzed. Thus, boxplots don't directly address the relationship between two categorical variables.
Therefore, creating crosstabs provides the most direct and informative method for exploring the relationship between the `dayofweek` and `binned_age` categorical features. It allows the machine learning specialist to understand the distribution of customers in different age groups on different days of the week, revealing valuable insights into product usage patterns.
Further Research:

Pandas crosstab function: https://pandas.pydata.org/docs/reference/api/pandas.crosstab.html

Contingency Table: https://en.wikipedia.org/wiki/Contingency_table

Question: 346

Exam Heist

A company needs to develop a model that uses a machine learning (ML) model for risk analysis. An ML engineer needs to evaluate the contribution each feature of a training dataset makes to the prediction of the target variable before the ML engineer selects features.

How should the ML engineer predict the contribution of each feature?

- Use the Amazon SageMaker Data Wrangler multicollinearity measurement features and the principal component analysis (PCA) algorithm to calculate the variance of the dataset along multiple directions in the feature space.
- Use an Amazon SageMaker Data Wrangler quick model visualization to find feature importance scores that are between 0.5 and 1. ✓
- Use the Amazon SageMaker Data Wrangler bias report to identify potential biases in the data related to feature engineering.
- Use an Amazon SageMaker Data Wrangler data flow to create and modify a data preparation pipeline. Manually add the feature scores.

Explanation:

The correct answer is **B. Use an Amazon SageMaker Data Wrangler quick model visualization to find feature importance scores that are between 0.5 and 1.**

Here's a detailed justification:

The task is to determine the contribution of each feature to the model's predictions, which is essentially feature importance analysis. Amazon SageMaker Data Wrangler's quick model visualization provides a convenient and intuitive way to achieve this.

SageMaker Data Wrangler's Quick Model Visualization: This feature within Data Wrangler builds a simple, explainable model (often a linear model or tree-based model) to predict the target variable. It then calculates and displays feature importance scores, indicating how much each feature contributes to the prediction. This is exactly what the ML engineer needs. These visualizations highlight the relative influence of each feature.

Feature Importance Scores: These scores typically range from 0 to 1 (or are normalized to this range). A score closer to 1 indicates that the feature has a significant impact on the model's predictions.

Let's examine why the other options are less suitable:

A. Use the Amazon SageMaker Data Wrangler multicollinearity measurement features and the principal component analysis (PCA) algorithm to calculate the variance of the dataset along multiple directions in the feature space. While PCA can reduce dimensionality and identify variance, it doesn't directly provide feature *importance* in relation to predicting the target variable in the same way feature importance scores do. Multicollinearity measures the correlation between independent variables, which is important for model stability, but doesn't directly address the feature's predictive power. This combination is useful for feature engineering, but doesn't offer a direct feature importance score.

C. Use the Amazon SageMaker Data Wrangler bias report to identify potential biases in the data related to feature engineering. Bias reports are crucial for fairness and ethical considerations, but they don't directly measure feature importance. Bias detection aims to uncover disparities or unfair influences related to specific features, which is different from determining how strongly a feature influences the target variable.

D. Use an Amazon SageMaker Data Wrangler data flow to create and modify a data preparation pipeline. Manually add the feature scores. While a data flow is essential for data preprocessing and transformation, manually adding feature scores would require the engineer to calculate or estimate these scores independently, which is less efficient and more prone to error than using Data Wrangler's built-in feature importance calculation within its quick model visualization. This approach is neither practical nor directly supported by Data Wrangler. Data Wrangler is designed to automate processes and using this tool to manual calculation defeats the purpose.

Therefore, using Data Wrangler's quick model visualization is the most efficient and direct way to determine the contribution of each feature to the prediction of the target variable.

Supporting links:

Amazon SageMaker Data Wrangler Documentation
Amazon SageMaker Feature Importance

Question: 347

Exam Heist

A company is building a predictive maintenance system using real-time data from devices on remote sites. There is no AWS Direct Connect connection or VPN connection between the sites and the company's VPC. The data needs to be ingested in real time from the devices into Amazon S3.

Transformation is needed to convert the raw data into clean .csv data to be fed into the machine learning (ML) model. The transformation needs to

happen during the ingestion process. When transformation fails, the records need to be stored in a specific location in Amazon S3 for human review. The raw data before transformation also needs to be stored in Amazon S3.

How should an ML specialist architect the solution to meet these requirements with the LEAST effort?

- Use Amazon Data Firehose with Amazon S3 as the destination. Configure Firehose to invoke an AWS Lambda function for data transformation. Enable source record backup on Firehose.
- Use Amazon Managed Streaming for Apache Kafka. Set up workers in Amazon Elastic Container Service (Amazon ECS) to move data from Kafka brokers to Amazon S3 while transforming it. Configure workers to store raw and unsuccessfully transformed data in different S3 buckets.
- Use Amazon Data Firehose with Amazon S3 as the destination. Configure Firehose to invoke an Apache Spark job in AWS Glue for data transformation. Enable source record backup and configure the error prefix. ✓
- Use Amazon Kinesis Data Streams in front of Amazon Data Firehose. Use Kinesis Data Streams with AWS Lambda to store raw data in Amazon S3. Configure Firehose to invoke a Lambda function for data transformation with Amazon S3 as the destination.

Explanation:

The best solution is **C. Use Amazon Data Firehose with Amazon S3 as the destination. Configure Firehose to invoke an Apache Spark job in AWS Glue for data transformation. Enable source record backup and configure the error prefix.**
Here's why:

Real-time Ingestion: Firehose is designed for real-time data ingestion into AWS services like S3, which aligns perfectly with the requirement of streaming data from remote devices.
Transformation during Ingestion: Firehose natively supports data transformation using AWS Lambda or AWS Glue. While Lambda is a valid option (as shown in option A and D), Glue with Apache Spark is generally better suited for complex transformations due to its distributed processing capabilities. In this case, the question states "Transformation is needed to convert the raw data into clean .csv data," which indicates that the transformations might involve complex data cleaning, filtering, or aggregation processes.
Error Handling: Firehose allows enabling source record backup to store raw data before transformation. Additionally, configuring an error prefix directs unsuccessfully transformed records to a specific location in S3 for human review.
Least Effort: Using Firehose with Glue minimizes the need for custom coding and infrastructure management compared to options like using Kafka and ECS.
Here's why other options are less suitable:

A: While Firehose with Lambda can handle basic transformations, it might not be as efficient or scalable for complex transformations as Glue with Spark. Also, Lambda has execution time limits that might become a bottleneck for handling bursts of data.
B: Kafka and ECS provide a powerful but more complex solution, involving significantly more setup and management overhead compared to Firehose and Glue. This violates the "least effort" requirement.
D: Introducing Kinesis Data Streams adds unnecessary complexity. Firehose can directly ingest the data from the devices without an intermediary streaming service.
Key Concepts:

Amazon Data Firehose: Fully managed service for real-time streaming data to destinations like S3, Redshift, and Elasticsearch. https://aws.amazon.com/kinesis/data-firehose/
AWS Glue: Fully managed ETL (extract, transform, load) service for preparing and loading data for analytics. https://aws.amazon.com/glue/
Apache Spark: A unified analytics engine for large-scale data processing. https://spark.apache.org/
AWS Lambda: Serverless compute service for running code without provisioning or managing servers. https://aws.amazon.com/lambda/
Justification Summary:
Option C utilizes Firehose and Glue, providing a manageable, scalable, and cost-effective solution. Firehose handles real-time data ingestion and error handling, while Glue, powered by Spark, performs robust data transformations. The built-in features of both services ensure the requirements are met with minimal effort.

Question: 348

Exam Heist

A company wants to use machine learning (ML) to improve its customer churn prediction model. The company stores data in an Amazon Redshift data warehouse.

A data science team wants to use Amazon Redshift machine learning (Amazon Redshift ML) to build a model and run predictions for new data directly within the data warehouse.

Which combination of steps should the company take to use Amazon Redshift ML to meet these requirements? (Choose three.)

- [x] Define the feature variables and target variable for the churn prediction model. ✓
- [ ] Use the SOL EXPLAIN_MODEL function to run predictions.
- [x] Write a CREATE MODEL SQL statement to create a model. ✓
- [ ] Use Amazon Redshift Spectrum to train the model.
- [ ] Manually export the training data to Amazon S3.
- [x] Use the SQL prediction function to run predictions. ✓

Explanation:
The correct answer is ACF because it outlines the core steps for using Amazon Redshift ML to build and use a churn prediction model directly within the Redshift data warehouse. Let's break down each choice:
A. Define the feature variables and target variable for the churn prediction model. This is a crucial preliminary step. Before creating any model, the data science team needs to identify which columns in the Redshift tables will serve as features (inputs) for predicting churn, and which column represents the target variable (whether or not a customer churned). This data exploration and feature engineering phase is essential for building a useful model.
C. Write a CREATE MODEL SQL statement to create a model. Amazon Redshift ML is designed to bring machine learning capabilities *into* the database environment. The core mechanism for this is the `CREATE MODEL` SQL command. This command specifies the type of model, the training data (from Redshift tables), the feature columns, the target column, and other parameters. This essentially initiates the model training process within Redshift.
https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_MODEL.html
F. Use the SQL prediction function to run predictions. Once the model is trained using the `CREATE MODEL` statement, the `PREDICT` function is used within SQL queries to generate predictions on new data residing in Redshift tables. This allows the company to seamlessly integrate predictions into their existing data workflows and dashboards.
https://docs.aws.amazon.com/redshift/latest/dg/r_PREDICT.html
Why the other options are incorrect:

B. Use the SOL EXPLAIN_MODEL function to run predictions. There isn't a standard or well-known SQL function named `EXPLAIN_MODEL` for running predictions in Amazon Redshift or other database systems.
D. Use Amazon Redshift Spectrum to train the model. While Redshift Spectrum is used for querying data in S3, it is *not* directly involved in training models with Redshift ML. The `CREATE MODEL` command trains the model using data within the Redshift cluster.
E. Manually export the training data to Amazon S3. A key advantage of Redshift ML is that it avoids the need to export data to external systems like S3 for training. The training data stays within the Redshift environment, enhancing security and efficiency. Data remains within Redshift as it's trained via `CREATE MODEL`.
In summary, the process involves defining the problem, building the model inside Redshift via SQL, and then scoring new data within the data warehouse using a prediction function within SQL.

Question: 349

Exam Heist

A company's machine learning (ML) team needs to build a system that can detect whether people in a collection of images are wearing the company's logo. The company has a set of labeled training data.

Which algorithm should the ML team use to meet this requirement?

- ( ) Principal component analysis (PCA)
- ( ) Recurrent neural network (RNN)
- ( ) K-nearest neighbors (k-NN)
- (•) Convolutional neural network (CNN) ✓

Explanation:
The task is to detect the presence of a specific logo in images. This is an image recognition problem, specifically object detection.

Convolutional Neural Networks (CNNs) are specifically designed for image-related tasks. They excel at learning spatial hierarchies of features from images, making them ideal for identifying objects within an image, such as a logo. CNNs use convolutional layers to extract features, pooling layers to reduce dimensionality, and fully connected layers for classification. They can be trained to recognize the specific logo even if it varies in size, orientation, or is partially occluded.
Principal Component Analysis (PCA) is a dimensionality reduction technique. It's useful for simplifying data and extracting the most important features, but it doesn't inherently identify objects in images. It is not suitable for object detection.

Recurrent Neural Networks (RNNs) are designed to handle sequential data like time series or text. They are not primarily used for image analysis or object detection. While some variants can be used with images they are not the primary option for an image classification problem.
K-Nearest Neighbors (k-NN) is a classification algorithm that classifies new data points based on the majority class of their nearest neighbors in the feature space. While k-NN could be used, it would be very inefficient with raw image data. Feature extraction would first need to be done (possibly using something like CNNs) to create vectors. CNNs are a better option as they do this feature extraction and classification within one end-to-end trained model.
Therefore, **CNNs** provide the most appropriate solution because they are specifically designed for image analysis and object detection tasks. They can effectively learn and recognize the company's logo in the images.
Authoritative Links:

Convolutional Neural Networks: https://www.tensorflow.org/tutorials/images/cnn
Object Detection Overview: https://paperswithcode.com/task/object-detection

Question: 350

Exam Heist

A data scientist uses Amazon SageMaker Data Wrangler to obtain a feature summary from a dataset that the data scientist imported from Amazon S3. The data scientist notices that the prediction power for a dataset feature has a score of 1.

What is the cause of the score?

- Target leakage occurred in the imported dataset. ✓
- The data scientist did not fine-tune the training and validation split.
- The SageMaker Data Wrangler algorithm that the data scientist used did not find an optimal model fit for each feature to calculate the prediction power.
- The data scientist did not process the features enough to accurately calculate prediction power.

Explanation:
The correct answer is A, Target leakage occurred in the imported dataset.
Here's why: A prediction power score of 1.0 from SageMaker Data Wrangler strongly indicates target leakage. Target leakage occurs when information about the target variable that would not be available at prediction time is inadvertently included in the training data. In other words, the feature being analyzed is almost perfectly correlated with the target.
Let's break down why the other options are less likely:

B. The data scientist did not fine-tune the training and validation split: While a poor train/validation split can affect model performance, it typically wouldn't result in a prediction power score of *exactly* 1. The score would likely be lower, indicating overfitting to the training data but not perfect correlation.
C. The SageMaker Data Wrangler algorithm that the data scientist used did not find an optimal model fit for each feature to calculate the prediction power: SageMaker Data Wrangler uses various algorithms to calculate prediction power. Even if the optimal model fit isn't achieved, it's improbable that *every* algorithm would yield a perfect prediction power score of 1 if leakage isn't present.
D. The data scientist did not process the features enough to accurately calculate prediction power: Insufficient feature processing could lead to a lower prediction power score or less reliable results. However, it wouldn't inherently cause a score of 1.0. Improper or incomplete feature engineering is the *cause* of Target Leakage, not just a predictor of accuracy.
Target leakage fundamentally breaks the principle that a model should only learn from information available at prediction time. A score of 1.0 signifies the model isn't learning generalized patterns but simply memorizing or directly using the target variable.
Example of Target Leakage: Imagine predicting whether a customer will purchase a product, and one of your features is "date_of_purchase." If you include this *future* information during training, the model will likely learn a near-perfect relationship, but the feature is useless at prediction time.
Authoritative Resources:

AWS Documentation on Feature Store and Data Preparation: The official AWS documentation helps understand how features are built and the dangers of not preparing them: https://docs.aws.amazon.com/sagemaker/latest/dg/feature-store.html
Google's discussion on Target Leakage: https://developers.google.com/machine-learning/data-prep/training-validation-test-split/leakage

Question: 351

Exam Heist

A data scientist is conducting exploratory data analysis (EDA) on a dataset that contains information about product suppliers. The dataset records the country where each product supplier is located as a two-letter text code. For example, the code for New Zealand is "NZ."

The data scientist needs to transform the country codes for model training. The data scientist must choose the solution that will result in the smallest increase in dimensionality. The solution must not result in any information loss.

Which solution will meet these requirements?

- ○ Add a new column of data that includes the full country name.
- ○ Encode the country codes into numeric variables by using similarity encoding.
- ○ Map the country codes to continent names.
- ○ Encode the country codes into numeric variables by using one-hot encoding. ✓

Explanation:
The question asks for a method to encode country codes for machine learning while minimizing dimensionality increase and avoiding information loss. Let's analyze each option:

A. Add a new column of data that includes the full country name. This increases dimensionality (adding a new feature) without directly addressing how to represent the data numerically for a machine learning model. While the information is retained, it doesn't solve the encoding problem.
B. Encode the country codes into numeric variables by using similarity encoding. Similarity encoding attempts to group similar values together based on some criteria (e.g., geographic proximity). This *could* reduce dimensionality compared to one-hot encoding, but it also introduces information loss because it collapses distinct countries into potentially overlapping categories. Further, defining the similarity metric adds complexity and potential bias.
C. Map the country codes to continent names. This dramatically reduces dimensionality, but it also loses significant information. Many distinct countries are grouped under a single continent label, leading to substantial information loss. This is unsuitable for the stated requirement of no information loss.
D. Encode the country codes into numeric variables by using one-hot encoding. One-hot encoding creates a new binary column for each unique country code. For example, if there are 100 unique country codes, one-hot encoding will add 100 new columns. This maintains all the original information since each country code is represented by a unique column. It's a standard practice for encoding categorical variables in machine learning. Although it increases dimensionality, it's the minimum increase possible while retaining all original information.
Therefore, the correct answer is D. One-hot encoding preserves all the information in the country codes while representing them numerically, which is essential for most machine learning algorithms. It avoids the information loss associated with similarity encoding or continent mapping. While adding new features (increased dimensionality) compared to the original single feature (country code), it does so in a lossless manner. This is a common and acceptable trade-off in many machine learning scenarios.
Supporting Links:

One-Hot Encoding: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
Categorical Data Encoding Techniques: https://contrib.scikit-learn.org/category_encoders/

Question: 352

Exam Heist

A data scientist is building a new model for an ecommerce company. The model will predict how many minutes it will take to deliver a package.

During model training, the data scientist needs to evaluate model performance.

Which metrics should the data scientist use to meet this requirement? (Choose two.)

- ☐ InferenceLatency
- ☑ Mean squared error (MSE) ✓
- ☑ Root mean squared error (RMSE) ✓
- ☐ Precision
- ☐ Accuracy

Explanation:
The problem requires selecting metrics to evaluate the performance of a model predicting package delivery time in minutes. This is a regression problem, as the target variable (delivery time) is continuous. Therefore, metrics appropriate for regression tasks should be selected.

Option B, Mean Squared Error (MSE), is a common metric for regression models. It calculates the average of the squared differences between predicted and actual values. A lower MSE indicates better model performance as it signifies smaller prediction errors.
Option C, Root Mean Squared Error (RMSE), is another widely used regression metric. It is the square root of the MSE. Taking the square root makes the metric interpretable in the same units as the target variable (minutes in this case). Similar to MSE, a lower RMSE signifies a better performing model. RMSE is often preferred over MSE because the error is expressed in the same units as the predicted variable, making it easier to understand and interpret.
Option A, Inference Latency, is relevant to the model's deployment and operational performance, indicating the time it takes to get a prediction. While important, it doesn't directly measure the model's accuracy.
Option D, Precision, and Option E, Accuracy, are classification metrics. Precision measures the proportion of positive identifications that were actually correct, while accuracy measures the overall correctness of the model's predictions. Since this is a regression problem, these classification metrics are inappropriate.
Therefore, MSE and RMSE (B and C) are the most suitable choices for evaluating the performance of the delivery time prediction model during training because they directly quantify the difference between the model's predictions and the actual delivery times.
Relevant links for further reading:

Mean Squared Error: https://en.wikipedia.org/wiki/Mean_squared_error
Root Mean Squared Error: https://en.wikipedia.org/wiki/Root-mean-square_deviation

Question: 353

Exam Heist

A machine learning (ML) specialist is developing a model for a company. The model will classify and predict sequences of objects that are displayed in a video. The ML specialist decides to use a hybrid architecture that consists of a convolutional neural network (CNN) followed by a classifier three-layer recurrent neural network (RNN).

The company developed a similar model previously but trained the model to classify a different set of objects. The ML specialist wants to save time by using the previously trained model and adapting the model for the current use case and set of objects.

Which combination of steps will accomplish this goal with the LEAST amount of effort? (Choose two.)

- [ ] Reinitialize the weights of the entire CNN. Retrain the CNN on the classification task by using the new set of objects.
- [ ] Reinitialize the weights of the entire network. Retrain the entire network on the prediction task by using the new set of objects.
- [ ] Reinitialize the weights of the entire RNN. Retrain the entire model on the prediction task by using the new set of objects.
- [x] Reinitialize the weights of the last fully connected layer of the CNN. Retrain the CNN on the classification task by using the new set of objects. ✓
- [x] Reinitialize the weights of the last layer of the RNN. Retrain the entire model on the prediction task by using the new set of objects. ✓

Explanation:
The question asks for the most efficient way to adapt a pre-trained CNN-RNN model for a new set of objects in a video sequence classification task. This scenario lends itself well to transfer learning.
Option D suggests reinitializing only the last fully connected layer of the CNN. This is crucial because this layer is directly responsible for mapping the CNN's learned features to the specific classes of objects in the previous dataset. Reinitializing this layer allows the CNN to learn a new mapping tailored to the current set of objects while retaining the valuable feature extraction capabilities learned in the original training.
Option E suggests reinitializing the last layer of the RNN. Because the RNN is responsible for handling the sequential aspect of the data and making predictions based on sequences of features extracted by the CNN, retraining the last layer of the RNN allows the model to adapt its prediction logic to the new set of objects while still leveraging the CNN's pre-trained feature extraction and the RNN's initial sequence understanding. This approach is more efficient than retraining the entire RNN from scratch.
Why other options are incorrect:

A: Reinitializing the entire CNN negates the benefits of transfer learning and wastes the previously learned feature extraction capabilities.
B: Reinitializing the entire network eliminates all benefits of pre-training, effectively starting from scratch, which is the opposite of minimizing effort.
C: Reinitializing the entire RNN is better than reinitializing the whole network, but it’s still more effort than just reinitializing the last layer.
Therefore, reinitializing the last layer of the CNN to learn new object classifications and reinitializing the last layer of the RNN to adapt to new sequence predictions of these objects offers the optimal balance between leveraging pre-trained knowledge and adapting to the new task with the least amount of effort.

Relevant links for further reading:

Transfer Learning: https://paperswithcode.com/method/transfer-learning
Fine-tuning CNNs: https://towardsdatascience.com/a-comprehensive-guide-to-fine-tuning-deep-learning-models-897d4a0d344b
Recurrent Neural Networks: https://www.tensorflow.org/guide/keras/rnn

Question: 354

Exam Heist

A company distributes an online multiple-choice survey to several thousand people. Respondents to the survey can select multiple options for each question.

A machine learning (ML) engineer needs to comprehensively represent every response from all respondents in a dataset. The ML engineer will use the dataset to train a logistic regression model.

Which solution will meet these requirements?

- Perform one-hot encoding on every possible option for each question of the survey. ✓
- Perform binning on all the answers each respondent selected for each question.
- Use Amazon Mechanical Turk to create categorical labels for each set of possible responses.
- Use Amazon Textract to create numeric features for each set of possible responses.

Explanation:
The correct solution is A: Perform one-hot encoding on every possible option for each question of the survey. Here's why:
The problem requires representing multiple-choice survey responses, where respondents can select multiple options per question, in a format suitable for a logistic regression model. Logistic regression, like many machine learning algorithms, works best with numerical input data.
One-hot encoding is a process of converting categorical variables into a numerical representation that machine learning algorithms can understand. In this scenario, each possible answer option for each question becomes a separate binary feature (column) in the dataset. If a respondent selected that option, the corresponding feature will have a value of 1; otherwise, it will be 0. This precisely captures the "multiple selections" aspect, because a respondent can have multiple columns with the value of 1 for a single question. This approach avoids introducing ordinality (ranking) where none exists since each option is treated as a distinct, independent feature.
Binning (option B) is generally used for numerical data to group values into ranges, which isn't suitable for representing selections from a list of pre-defined options. Using Amazon Mechanical Turk (option C) to create categorical labels is unnecessary and introduces subjective bias. The original data provides the categories already. Amazon Textract (option D) is designed for extracting text and data from scanned documents, which is irrelevant to this problem where the survey responses are already available in a structured format.
One-hot encoding allows a logistic regression model to correctly learn relationships between the selected survey options and the target variable. The resulting dataset will have a clear, well-defined structure which allows logistic regression to accurately calculate probabilities and make predictions. The other options do not efficiently or accurately convert multi-select data for use within a logistic regression model.For further research on one-hot encoding, refer to these resources:

One-Hot Encoding: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
Categorical Data Encoding: https://contrib.scikit-learn.org/category_encoders/onehot.html

Question: 355

Exam Heist

A manufacturing company stores production volume data in a PostgreSQL database.

The company needs an end-to-end solution that will give business analysts the ability to prepare data for processing and to predict future production volume based the previous year's production volume. The solution must not require the company to have coding knowledge.

Which solution will meet these requirements with the LEAST effort?

- Use AWS Database Migration Service (AWS DMS) to transfer the data from the PostgreSQL database to an Amazon S3 bucket. Create an Amazon EMR duster to read the S3 bucket and perform the data preparation. Use Amazon SageMaker Studio for the prediction modeling.
- Use AWS Glue DataBrew to read the data that is in the PostgreSQL database and to perform the data preparation. Use Amazon SageMaker Canvas for the prediction modeling. ✓

- Use AWS Database Migration Service (AWS DMS) to transfer the data from the PostgreSQL database to an Amazon S3 bucket. Use AWS Glue to read the data in the S3 bucket and to perform the data preparation. Use Amazon SageMaker Canvas for the prediction modeling.
- Use AWS Glue DataBrew to read the data that is in the PostgreSQL database and to perform the data preparation. Use Amazon SageMaker Studio for the prediction modeling.

Explanation:

The correct answer is **B. Use AWS Glue DataBrew to read the data that is in the PostgreSQL database and to perform the data preparation. Use Amazon SageMaker Canvas for the prediction modeling.**

Here's why:

Least Effort and No-Code Requirement: The key requirement is to minimize coding and effort for business analysts. Amazon SageMaker Canvas is specifically designed for non-technical users to build machine learning models without writing code. AWS Glue DataBrew is also a no-code data preparation tool.

Data Preparation: AWS Glue DataBrew provides a visual interface for data cleaning, normalization, and transformation, directly addressing the data preparation needs from the PostgreSQL database. It can connect directly to various data sources, including PostgreSQL.

Prediction Modeling: SageMaker Canvas allows business analysts to build predictive models using a drag-and-drop interface. Users can select the target variable and the Canvas will provide recommendations for different model types to use, then trains and deploys the model automatically.

Why other options are less suitable:

A: While Amazon EMR can perform data preparation, it typically requires coding knowledge (e.g., using Spark or Python). SageMaker Studio is more aimed at data scientists and requires some coding experience.

C: Using AWS Glue for data preparation is a valid approach, but it usually requires some coding knowledge. SageMaker Canvas, as mentioned, is the best option to adhere the no-code requirement.

D: AWS Glue DataBrew suits the data preparation needs, but SageMaker Studio is designed for users with some coding knowledge.

Supporting Links:

AWS Glue DataBrew: https://aws.amazon.com/glue/databrew/

Amazon SageMaker Canvas: https://aws.amazon.com/sagemaker/canvas/

Question: 356

Exam Heist

A data scientist needs to create a model for predictive maintenance. The model will be based on historical data to identify rare anomalies in the data.

The historical data is stored in an Amazon S3 bucket. The data scientist needs to use Amazon SageMaker Data Wrangler to ingest the data. The data scientist also needs to perform exploratory data analysis (EDA) to understand the statistical properties of the data.

Which solution will meet these requirements with the LEAST amount of compute resources?

- Import the data by using the None option.
- Import the data by using the Stratified option.
- Import the data by using the First K option. Infer the value of K from domain knowledge.
- Import the data by using the Randomized option. Infer the random size from domain knowledge. ✓

Explanation:

The correct answer is **D. Import the data by using the Randomized option. Infer the random size from domain knowledge.**

Here's a detailed justification:

The problem requires using SageMaker Data Wrangler for EDA and anomaly detection in a large historical dataset stored in S3, focusing on minimizing compute resources.

A. Import the data by using the None option: This option implies importing the entire dataset, which would consume the most compute resources, directly contradicting the requirement to minimize them. EDA on the entire dataset can be computationally expensive.

B. Import the data by using the Stratified option: Stratified sampling ensures that subgroups (strata) within the dataset are represented proportionally in the sample. While useful for maintaining class balance, especially in imbalanced datasets, it doesn't necessarily reduce compute resources as effectively as randomized sampling, especially if the whole strata are large.

C. Import the data by using the First K option: This option involves importing the first *K* records. This might not be representative of the overall dataset, particularly if the data is sorted or ordered in a specific way. EDA based on the first K records might lead to biased or inaccurate insights. It does not consider the rarity of anomalies, and it can lead to highly biased EDA and inaccurate modelling.
D. Import the data by using the Randomized option: This option selects a random subset of the data. This method is best for minimizing resources because it significantly reduces the data volume ingested while still providing a statistically representative sample for initial EDA. By inferring a suitable random sample size from domain knowledge, the data scientist can balance resource consumption with the need to capture the characteristics of the data, including potential anomalies. Because the aim is anomaly detection, the model needs to examine as much of the dataset as possible with the least resources. A random sample that takes data from all segments of the dataset would be more effective at anomaly detection with EDA than the other options.
By using a randomized sample, the data scientist can perform initial EDA to understand data distributions and identify potential anomalies without processing the entire dataset. Subsequent more targeted analysis can then be performed if needed. Domain knowledge helps in determining the appropriate sample size that balances representativeness and resource efficiency.
Relevant Cloud Computing Concepts:

Amazon SageMaker Data Wrangler: A fully managed service for data preparation that allows you to quickly and easily explore, clean, and prepare data for machine learning.
Exploratory Data Analysis (EDA): An approach to analyzing data sets to summarize their main characteristics, often with visual methods.
Data Sampling: A statistical analysis technique used to select, manipulate and analyze a representative subset of data points in order to identify patterns and trends in the larger data set being examined.
Cost Optimization: A critical factor in cloud computing that involves selecting the most efficient resources and configurations to minimize expenses.
Authoritative Links:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Data Sampling Techniques: https://www.tableau.com/data-insights/statistical-terms/sampling

Question: 357

Exam Heist

An ecommerce company has observed that customers who use the company's website rarely view items that the website recommends to customers. The company wants to recommend items to customers that customers are more likely to want to purchase.

Which solution will meet this requirement in the SHORTEST amount of time?

- Host the company's website on Amazon EC2 Accelerated Computing instances to increase the website response speed.
- Host the company's website on Amazon EC2 GPU-based instances to increase the speed of the website's search tool.
- Integrate Amazon Personalize into the company's website to provide customers with personalized recommendations. ✓
- Use Amazon SageMaker to train a Neural Collaborative Filtering (NCF) model to make product recommendations.

Explanation:
The best solution for quickly providing personalized product recommendations is **C. Integrate Amazon Personalize into the company's website to provide customers with personalized recommendations.**
Here's why:
Amazon Personalize is a fully managed machine learning service specifically designed for real-time personalized recommendations. It removes the complexities of building, training, and deploying recommendation systems. Its "out-of-the-box" nature and tight integration with other AWS services makes it the quickest path to implementing a personalized recommendation engine.
Option A (EC2 Accelerated Computing instances) and B (EC2 GPU-based instances) focus on improving website performance and search speed, respectively. While these are valuable optimizations, they don't directly address the core issue of personalized recommendations. Faster page loading or search won't inherently improve the relevance of recommended items.
Option D (SageMaker and NCF model) is a more involved approach. While SageMaker is powerful, it requires significant effort to build, train, deploy, and manage the model. This includes data preparation, algorithm selection, hyperparameter tuning, and continuous monitoring. This is a time-consuming process compared to using a managed service like Personalize.Furthermore, the NCF model is a specific algorithm for collaborative filtering, which might not be the best choice for all types of product recommendation problems. Personalize offers a variety of algorithms and automatically selects the best one based on the dataset.

In summary, Amazon Personalize excels in speed of implementation and management, providing the functionality specifically required (personalized recommendations) with minimal development effort.It enables businesses to leverage ML without the high complexity of managing the ML infrastructure by implementing personalized recommendations quickly.
Relevant Links:

Amazon Personalize: https://aws.amazon.com/personalize/
Amazon SageMaker: https://aws.amazon.com/sagemaker/

Question: 358

Exam Heist

A machine learning (ML) engineer is preparing a dataset for a classification model. The ML engineer notices that some continuous numeric features have a significantly greater value than most other features. A business expert explains that the features are independently informative and that the dataset is representative of the target distribution.

After training, the model's inferences accuracy is lower than expected.

Which preprocessing technique will result in the GREATEST increase of the model's inference accuracy?

- Normalize the problematic features. ✓
- Bootstrap the problematic features.
- Remove the problematic features.
- Extrapolate synthetic features.

Explanation:
The situation describes features with significantly differing magnitudes, negatively impacting model performance. Choice A, normalizing the problematic features, is the best solution. Normalization scales numeric features to a standard range, typically between 0 and 1 or around a zero mean with unit variance. This brings all features onto a similar scale, preventing features with larger values from dominating the model's learning process. This is especially crucial for algorithms sensitive to feature scaling, like distance-based algorithms (e.g., k-NN, support vector machines) and gradient descent-based algorithms (e.g., neural networks).
Bootstrapping (B) is a resampling technique used to estimate the sampling distribution of a statistic. It won't address the issue of feature scaling. Removing the informative features (C) will directly decrease the model's accuracy, which is the opposite of the goal. Extrapolating synthetic features (D) could introduce bias if not done carefully and doesn't directly address the scaling problem. While feature engineering can sometimes improve performance, normalizing the existing, informative features is a more direct and generally safer initial approach in this scenario. Normalization helps ensure that the model gives each feature appropriate consideration, leading to improved learning and generalization.Therefore, normalization directly addresses the problem of disparate feature scales, leading to the highest expected increase in model inference accuracy.
Relevant links:

Feature Scaling
Normalization vs Standardization

Question: 359

Exam Heist

A manufacturing company produces 100 types of steel rods. The rod types have varying material grades and dimensions. The company has sales data for the steel rods for the past 50 years.

A data scientist needs to build a machine learning (ML) model to predict future sales of the steel rods.

Which solution will meet this requirement in the MOST operationally efficient way?

- Use the Amazon SageMaker DeepAR forecasting algorithm to build a single model for all the products. ✓
- Use the Amazon SageMaker DeepAR forecasting algorithm to build separate models for each product.
- Use Amazon SageMaker Autopilot to build a single model for all the products.
- Use Amazon SageMaker Autopilot to build separate models for each product.

Explanation:
The most operationally efficient solution is to use Amazon SageMaker DeepAR to build a single model for all products (Option A). Here's why:

DeepAR's Strength: DeepAR is a supervised learning algorithm specifically designed for time-series forecasting problems with numerous related time series. It excels in scenarios where you have multiple similar products (in this case, different types of steel rods) and can learn from the collective history of all products. This allows it to leverage patterns and dependencies across all the time series, improving forecast accuracy.
Efficiency: Building a single DeepAR model is more efficient than building separate models for each product (Option B). Training and managing 100 individual models requires significantly more computational resources, time, and effort. A single model streamlines the process and reduces operational overhead.
SageMaker Autopilot Limitation: SageMaker Autopilot (Options C and D) automates the model building process but typically focuses on finding the best single model for a single dataset or target variable. While it can be useful, it might not be the best approach for a multivariate time series forecasting problem.
Cross-Series Learning: DeepAR can learn dependencies and similarities across different steel rod types. For instance, if there's a general economic trend affecting the demand for all steel products, DeepAR can capture this common pattern and apply it to the forecasts for individual products. Autopilot built on separate models will not be able to see these correlations between products.
Cold Start Problem: If a new steel rod type is introduced with limited historical data, DeepAR can leverage the information from existing products to make a more accurate forecast. Individual models (Options B and D) would struggle with the "cold start" problem due to insufficient data.
Operational Overhead: Maintaining and monitoring 100 separate Autopilot models will be operationally complex, especially if regular retraining and updates are needed.
Scalability: A single DeepAR model can be more easily scaled and deployed compared to managing numerous individual models.
Therefore, a single DeepAR model provides the best combination of accuracy, efficiency, and manageability for this forecasting problem.
Supporting Links:

Amazon SageMaker DeepAR: https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html
Amazon SageMaker Autopilot: https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html

Question: 360

Exam Heist

A machine learning (ML) specialist is building a credit score model for a financial institution. The ML specialist has collected data for the previous 3 years of transactions and third-party metadata that is related to the transactions.

After the ML specialist builds the initial model, the ML specialist discovers that the model has low accuracy for both the training data and the test data. The ML specialist needs to improve the accuracy of the model.

Which solutions will meet this requirement? (Choose two.)

- [x] Increase the number of passes on the existing training data. Perform more hyperparameter tuning. ✓
- [ ] Increase the amount of regularization. Use fewer feature combinations.
- [x] Add new domain-specific features. Use more complex models. ✓
- [ ] Use fewer feature combinations. Decrease the number of numeric attribute bins.
- [ ] Decrease the amount of training data examples. Reduce the number of passes on the existing training data.

Explanation:
The scenario describes a situation where a machine learning model exhibits low accuracy on both training and test data, indicating **underfitting**. Underfitting occurs when the model is too simple to capture the underlying patterns in the data. To address this, we need to increase the model's complexity or provide it with more relevant information.
Option A, "Increase the number of passes on the existing training data. Perform more hyperparameter tuning," is partially correct. Increasing the number of passes, often referred to as increasing the number of epochs, allows the model to learn more from the existing data. However, doing this alone is insufficient. Hyperparameter tuning is crucial for finding the optimal settings for the model, which can significantly improve its performance and prevent overfitting as well as addressing underfitting.
Option C, "Add new domain-specific features. Use more complex models," is also correct. Adding new domain-specific features provides the model with additional information that can help it to better understand the relationships within the data. Using more complex models, such as moving from a linear regression to a neural network, increases the model's capacity to learn complex patterns, thus potentially resolving underfitting.

Option B, "Increase the amount of regularization. Use fewer feature combinations," is incorrect. Increasing regularization is a technique used to prevent overfitting, which is the opposite of underfitting. Using fewer feature combinations also simplifies the model, which can exacerbate the underfitting problem.
Option D, "Use fewer feature combinations. Decrease the number of numeric attribute bins," is incorrect for similar reasons as option B. Both actions simplify the model, which will worsen the underfitting problem. Decreasing the number of numeric attribute bins reduces the granularity of the data, potentially losing important information.
Option E, "Decrease the amount of training data examples. Reduce the number of passes on the existing training data," is incorrect. Reducing the amount of training data and the number of passes will further limit the model's ability to learn and will only worsen the underfitting problem.
Therefore, the most effective solutions to address underfitting are to increase the model's complexity or provide it with more relevant information through new features and thorough hyperparameter tuning.
Supporting Links:

Underfitting and Overfitting: https://www.ibm.com/cloud/learn/overfitting-and-underfitting
Hyperparameter Tuning: https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning.html

Question: 361

Exam Heist

A data scientist uses Amazon SageMaker to perform hyperparameter tuning for a prototype machine leaming (ML) model. The data scientist's domain knowledge suggests that the hyperparameter is highly sensitive to changes.

The optimal value, x, is in the 0.5 < x < 1.0 range. The data scientist's domain knowledge suggests that the optimal value is close to 1.0.

The data scientist needs to find the optimal hyperparameter value with a minimum number of runs and with a high degree of consistent tuning conditions.

Which hyperparameter scaling type should the data scientist use to meet these requirements?

- Auto scaling
- Linear scaling
- Logarithmic scaling
- Reverse logarithmic scaling ✓

Explanation:
The data scientist aims to efficiently find the optimal hyperparameter value (x) within the range 0.5 < x < 1.0, expecting it to be close to 1.0 and highly sensitive to changes. The goal is to minimize the number of tuning runs while maintaining consistent conditions. Different hyperparameter scaling types offer varying search granularities.
Linear scaling would distribute the search space evenly, which is not ideal since the data scientist suspects the optimal value is near 1.0. Auto scaling lets SageMaker choose, which doesn't leverage the available domain knowledge. Logarithmic scaling is suitable when the hyperparameter's effect has a logarithmic relationship with the objective metric, concentrating the search towards smaller values. This is unsuitable as the data scientist expects the ideal value to be near 1.0, the higher end of the allowed range.
Reverse logarithmic scaling concentrates the search towards larger values, making it the appropriate choice. It provides finer granularity towards 1.0 and coarser granularity toward 0.5, efficiently focusing the search where the optimal value is likely to be. The sensitivity requirement means a small change in x near 1.0 can significantly impact the model's performance. Reverse log scaling enables a more refined search in this critical region, improving the likelihood of finding the optimal value with fewer iterations and ensuring consistent, granular tuning where it matters most.
Refer to the AWS documentation on hyperparameter tuning for further insights into hyperparameter scaling.https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-define-ranges.html

Question: 362

Exam Heist

A data scientist uses Amazon SageMaker Data Wrangler to analyze and visualize data. The data scientist wants to refine a training dataset by selecting predictor variables that are strongly predictive of the target variable. The target variable correlates with other predictor variables.

The data scientist wants to understand the variance in the data along various directions in the feature space.

Which solution will meet these requirements?

- Use the SageMaker Data Wrangler multicollinearity measurement features with a variance inflation factor (VIF) score. Use the VIF score as a measurement of how closely the variables are related to each other.

- Use the SageMaker Data Wrangler Data Quality and Insights Report quick model visualization to estimate the expected quality of a model that is trained on the data.
- Use the SageMaker Data Wrangler multicollinearity measurement features with the principal component analysis (PCA) algorithm to provide a feature space that includes all of the predictor variables. ✓
- Use the SageMaker Data Wrangler Data Quality and Insights Report feature to review features by their predictive power.

Explanation:

The correct answer is **C**. Here's why:

The data scientist needs to address two primary requirements: 1) select predictor variables strongly predictive of the target and 2) understand the variance in the data's feature space.

Option C directly addresses both requirements using SageMaker Data Wrangler's features. Multicollinearity measurement identifies correlated predictor variables. PCA (Principal Component Analysis) is a dimensionality reduction technique. It transforms the original features into a new set of uncorrelated features called principal components. These components are ordered by the amount of variance they explain, allowing the data scientist to understand the data's variance along various directions in the feature space. By retaining a sufficient number of principal components, the data scientist can represent most of the variance in the original data while reducing dimensionality and mitigating multicollinearity. The PCA output includes all predictor variables, transforming them, rather than removing them outright, which satisfies the requirement to understand variance along *all* the features.

Option A is incorrect because while VIF measures multicollinearity, it doesn't address the need to understand variance along various directions in the feature space. VIF identifies multicollinearity, but it doesn't provide a new feature space or a way to visualize or understand the variance distribution.

Option B is incorrect. Data Quality and Insights Reports' quick model visualization gives insight into the model quality but does not help in understanding the variance in the data along various directions in the feature space. It's more about model performance than feature understanding.

Option D is insufficient on its own. While the Data Quality and Insights Report can show predictive power, it doesn't explicitly provide a mechanism to understand the variance distribution in the feature space or to address multicollinearity directly.

Therefore, only option C fully addresses the requirements by leveraging PCA for variance understanding and multicollinearity assessment using Data Wrangler's capabilities.

Further research:

Amazon SageMaker Data Wrangler: https://aws.amazon.com/sagemaker/data-wrangler/
Principal Component Analysis (PCA): https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
Multicollinearity: https://en.wikipedia.org/wiki/Multicollinearity

Question: 363

Exam Heist

A business to business (B2B) ecommerce company wants to develop a fair and equitable risk mitigation strategy to reject potentially fraudulent transactions. The company wants to reject fraudulent transactions despite the possibility of losing some profitable transactions or customers.

Which solution will meet these requirements with the LEAST operational effort?

- Use Amazon SageMaker to approve transactions only for products the company has sold in the past.
- Use Amazon SageMaker to train a custom fraud detection model based on customer data.
- Use the Amazon Fraud Detector prediction API to approve or deny any activities that Fraud Detector identifies as fraudulent. ✓
- Use the Amazon Fraud Detector prediction API to identify potentially fraudulent activities so the company can review the activities and reject fraudulent transactions.

Explanation:

The correct answer is **C. Use the Amazon Fraud Detector prediction API to approve or deny any activities that Fraud Detector identifies as fraudulent.**

Here's why:

Amazon Fraud Detector (AFD) is purpose-built for fraud detection. It offers pre-trained models and the ability to customize models with your own data, minimizing operational overhead compared to building a model from scratch in SageMaker.

Least Operational Effort: AFD's prediction API directly provides a decision (approve/deny), fulfilling the requirement of automating the risk mitigation strategy without manual intervention. This is the most automated option, minimizing operational burden.

Prioritizing Rejection of Fraudulent Transactions (Even at the Cost of Losing Some Good Ones): The question emphasizes rejecting fraudulent transactions, even if it means losing some legitimate business. Configuring AFD to automatically deny

transactions identified as fraudulent achieves this objective directly.
Option A (SageMaker for Past Products) is too restrictive. Limiting transactions to past products is overly simplistic and won't capture new fraud patterns or legitimate purchases of new products.
Option B (Custom SageMaker Model) involves significant operational overhead. Building, training, deploying, and maintaining a custom fraud detection model in SageMaker requires significant data science and engineering effort, increasing operational complexity. While it offers customization, it's more effort than AFD.
Option D (AFD for Identification, Manual Review) adds manual review, contradicting the "least operational effort" requirement. This increases operational overhead because it requires human intervention to review and decide on potentially fraudulent transactions. It does not immediately reject transactions, defeating the purpose.
In Summary:
Amazon Fraud Detector directly addresses the need for a fair, equitable, and automated risk mitigation strategy for fraud detection with minimal operational burden. It allows for immediate rejection of transactions AFD identifies as fraudulent, aligning with the company's preference for aggressively preventing fraud, even at the expense of some potentially profitable transactions. SageMaker would be a better solution if the company had unique fraud scenarios that Amazon Fraud Detector does not support out-of-the-box.
Authoritative Links:

Amazon Fraud Detector: https://aws.amazon.com/fraud-detector/
Amazon SageMaker: https://aws.amazon.com/sagemaker/

Question: 364

Exam Heist

A data scientist needs to develop a model to detect fraud. The data scientist has less data for fraudulent transactions than for legitimate transactions.

The data scientist needs to check for bias in the model before finalizing the model. The data scientist needs to develop the model quickly.

Which solution will meet these requirements with the LEAST operational overhead?

- Process and reduce bias by using the synthetic minority oversampling technique (SMOTE) in Amazon EMR. Use Amazon SageMaker Studio Classic to develop the model. Use Amazon Augmented AI (Amazon A2I) to check the model for bias before finalizing the model.
- Process and reduce bias by using the synthetic minority oversampling technique (SMOTE) in Amazon EMR. Use Amazon SageMaker Clarify to develop the model. Use Amazon Augmented AI (Amazon A2I) to check the model for bias before finalizing the model.
- Process and reduce bias by using the synthetic minority oversampling technique (SMOTE) in Amazon SageMaker Studio. Use Amazon SageMaker JumpStart to develop the model. Use Amazon SageMaker Clarify to check the model for bias before finalizing the model. ✓
- Process and reduce bias by using an Amazon SageMaker Studio notebook. Use Amazon SageMaker JumpStart to develop the model. Use Amazon SageMaker Model Monitor to check the model for bias before finalizing the model.

Explanation:
The correct answer is C because it provides a streamlined and efficient solution for addressing the data imbalance and bias detection requirements with minimal operational overhead. Let's break down why:

SMOTE in Amazon SageMaker Studio: Performing SMOTE (Synthetic Minority Oversampling Technique) directly within SageMaker Studio avoids the need for a separate EMR cluster, reducing operational complexity and cost. SageMaker Studio provides a managed environment for data processing and model development.
Amazon SageMaker JumpStart: JumpStart provides pre-built models and notebooks for common machine learning tasks. Leveraging JumpStart allows the data scientist to quickly get started with a pre-trained model suitable for fraud detection, accelerating the model development process, which aligns with the requirement to develop the model quickly.
Amazon SageMaker Clarify: SageMaker Clarify is specifically designed to detect and mitigate bias in machine learning models and datasets. Using Clarify directly within the SageMaker ecosystem avoids the need for a separate bias detection service like A2I in the context of bias assessment.
Option A is less optimal because it uses Amazon EMR for SMOTE which increases operational overhead and costs compared to doing it directly in SageMaker Studio. It also proposes the use of SageMaker Studio Classic which is less featured than SageMaker Studio for modern ML development.
Option B uses SMOTE in EMR which increases complexity. Although it leverages SageMaker Clarify, it still uses Amazon A2I which is less efficient for automated bias analysis.
Option D utilizes SageMaker Model Monitor. While Model Monitor is useful for detecting data drift and model performance degradation in production, SageMaker Clarify is the appropriate choice for pre-deployment bias assessment. Furthermore, using a simple SageMaker Studio notebook for SMOTE is less efficient than leveraging the built-in functionalities in option C.
In conclusion, Option C provides the most efficient and integrated solution for bias detection and mitigation in the SageMaker environment with the least operational overhead.

Supporting Links:

Amazon SageMaker Clarify: https://aws.amazon.com/sagemaker/clarify/
Amazon SageMaker JumpStart: https://aws.amazon.com/sagemaker/jumpstart/
Synthetic Minority Oversampling Technique (SMOTE): Search for research papers and articles regarding SMOTE.
Amazon SageMaker Studio: https://aws.amazon.com/sagemaker/studio/

Question: 365

Exam Heist

A company has 2,000 retail stores. The company needs to develop a new model to predict demand based on holidays and weather conditions. The model must predict demand in each geographic area where the retail stores are located.

Before deploying the newly developed model, the company wants to test the model for 2 to 3 days. The model needs to be robust enough to adapt to supply chain and retail store requirements.

Which combination of steps should the company take to meet these requirements with the LEAST operational overhead? (Choose two.)

- [ ] Develop the model by using the Amazon Forecast Prophet model.
- [x] Develop the model by using the Amazon Forecast holidays featurization and weather index. ✓
- [x] Deploy the model by using a canary strategy that uses Amazon SageMaker and AWS Step Functions. ✓
- [ ] Deploy the model by using an A/B testing strategy that uses Amazon SageMaker Pipelines.
- [ ] Deploy the model by using an A/B testing strategy that uses Amazon SageMaker and AWS Step Functions.

Explanation:
Here's a breakdown of why options B and C are the most suitable choices for the problem scenario, focusing on minimizing operational overhead:
Option B: Develop the model by using the Amazon Forecast holidays featurization and weather index.

Amazon Forecast's Relevance: The problem statement specifically mentions predicting demand based on holidays and weather conditions. Amazon Forecast is designed for time-series forecasting, making it a natural fit for demand prediction.
Holidays Featurization: Forecast offers built-in holidays featurization. This automatically incorporates the impact of holidays (e.g., Black Friday, Christmas) into the model without requiring extensive manual feature engineering. This reduces development time and complexity.
Weather Index: While Forecast doesn't natively integrate with all weather data sources, you can incorporate a weather index (or features) from external weather APIs (e.g., AccuWeather, OpenWeatherMap) into your dataset and ingest that to Forecast. This handles the weather condition aspect efficiently.
Low Operational Overhead: Using Forecast's built-in capabilities minimizes the amount of custom code and infrastructure you need to manage, lowering operational overhead.
Option C: Deploy the model by using a canary strategy that uses Amazon SageMaker and AWS Step Functions.

Canary Deployment Benefits: A canary deployment involves deploying the new model to a small subset of the 2000 retail stores (the "canary"). This allows you to test the model's performance in a real-world environment with minimal risk. Any issues will only affect a small number of stores.
SageMaker and Step Functions Combination: SageMaker provides the environment to host and serve the deployed model. Step Functions provides the orchestration logic to gradually increase traffic to the new model (the canary) while monitoring its performance using CloudWatch metrics. This automated approach is ideal for gradual rollouts.
Meeting Testing Requirements: The canary strategy allows you to test the model for the specified 2-3 days on a representative sample of stores, ensuring it adapts to supply chain and retail requirements before full deployment.
Reduced Risk: By carefully monitoring the canary's performance, you can identify and address potential issues before they affect a larger portion of your retail stores.
Lower Overhead than A/B Testing: A/B testing would be more complex in this scenario. Canary deployments give immediate feedback on critical performance metrics. A/B testing can add more overhead because more traffic is dedicated to the new model and involves more granular performance analysis of different variants.
Why Other Options Are Less Suitable:

A: Develop the model by using the Amazon Forecast Prophet model. Prophet is just one of the models offered inside Amazon Forecast. This doesn't provide the same benefits as the holidays featurization offered inside Amazon Forecast.
D: Deploy the model by using an A/B testing strategy that uses Amazon SageMaker Pipelines. While A/B testing is a valid deployment strategy, it is better suited for comparing two existing models rather than an initial test of a new model. A/B testing requires more advanced statistical analysis. Canary is less risky to implement and easier to monitor for initial validation.

E: Deploy the model by using an A/B testing strategy that uses Amazon SageMaker and AWS Step Functions. Similar to D, A/B testing increases overhead compared to a canary strategy for initial testing.
Authoritative Links:

Amazon Forecast: https://aws.amazon.com/forecast/
Amazon SageMaker: https://aws.amazon.com/sagemaker/
AWS Step Functions: https://aws.amazon.com/step-functions/
Canary Deployments: Search for "canary deployment AWS" for architectural examples.
In summary, Amazon Forecast with holiday featurization allows for efficient model development, and a canary deployment orchestrated with SageMaker and Step Functions allows for controlled and low-risk deployment and validation. This combination minimizes the operational burden.

Question: 366

Exam Heist

A finance company has collected stock return data for 5,000 publicly traded companies. A financial analyst has a dataset that contains 2,000 attributes for each company. The financial analyst wants to use Amazon SageMaker to identify the top 15 attributes that are most valuable to predict future stock returns.

Which solution will meet these requirements with the LEAST operational overhead?

- Use the linear leaner algorithm in SageMaker to train a linear regression model to predict the stock returns. Identify the most predictive features by ranking absolute coefficient values.
- Use random forest regression in SageMaker to train a model to predict the stock returns. Identify the most predictive features based on Gini importance scores.
- Use an Amazon SageMaker Data Wrangler quick model visualization to predict the stock returns. Identify the most predictive features based on the quick mode's feature importance scores.
- Use Amazon SageMaker Autopilot to build a regression model to predict the stock returns. Identify the most predictive features based on an Amazon SageMaker Clarify report. ✓

Explanation:
The correct answer is D because it leverages SageMaker's Autopilot and Clarify, providing a streamlined and comprehensive solution for feature importance identification with minimal manual intervention. Autopilot automatically explores different models and preprocessing techniques, effectively addressing the analyst's goal of building a regression model to predict stock returns without extensive manual model selection and tuning. SageMaker Clarify then generates a report that details feature importance, revealing the top 15 attributes contributing most to the model's predictive power. This combination minimizes operational overhead by automating model selection, training, and feature importance analysis.
Option A, while functional, requires manual model selection (linear learner) and might not capture non-linear relationships between features and stock returns. Additionally, ranking absolute coefficient values might not fully represent feature importance, especially with multicollinearity.
Option B, random forest regression, could also work, but lacks the automated model selection and comprehensive reporting features of Autopilot and Clarify. Gini importance scores are useful but are not as easily interpretable and detailed as a Clarify report.
Option C using SageMaker Data Wrangler is useful for initial data exploration and visualization, but it is primarily designed for data preparation and less focused on building a predictive model and extracting feature importance in an automated and reproducible way, increasing the operational overhead. While it provides quick model visualizations, this might lack the rigor and depth required for identifying the top 15 attributes with confidence.
Therefore, utilizing Autopilot to build the regression model and then employing Clarify to generate a feature importance report is the most automated, comprehensive, and efficient approach for identifying the top 15 most valuable features, thus minimizing operational overhead.
Supporting Links:

Amazon SageMaker Autopilot: https://aws.amazon.com/sagemaker/autopilot/
Amazon SageMaker Clarify: https://aws.amazon.com/sagemaker/clarify/

Question: 367

Exam Heist

A company is using a machine learning (ML) model to recommend products to customers. An ML specialist wants to analyze the data for the most popular recommendations in four dimensions.

The ML specialist will visualize the first two dimensions as coordinates. The third dimension will be visualized as color. The ML specialist will use size

to represent the fourth dimension in the visualization

Which solution will meet these requirements?

- Use the Amazon SageMaker Data Wrangler bar chart feature. Use Group By to represent the third and fourth dimensions.
- Use the Amazon SageMaker Canvas box plot visualization Use color and fill pattern to represent the third and fourth dimensions
- Use the Amazon SageMaker Data Wrangler histogram feature Use color and fill pattern to represent the third and fourth dimensions
- Use the Amazon SageMaker Canvas scatter plot visualization Use scatter point size and color to represent the third and fourth dimensions ✓

Explanation:

D. Use the Amazon SageMaker Canvas scatter plot visualization.

Scatter plots allow for effective representation of multiple dimensions by using scatter point size and color to encode the third and fourth dimensions. This technique helps visualize relationships between variables while maintaining clarity.

Question: 368

Exam Heist

A clothing company is experimenting with different colors and materials for its products. The company stores the entire sales history of all its products in Amazon S3. The company is using custom-built exponential smoothing (ETS) models to forecast demand for its current products. The company needs to forecast the demand for a new product variation that the company will launch soon.

Which solution will meet these requirements?

- Train a custom ETS model.
- Train an Amazon SageMaker DeepAR model. ✓
- Train an Amazon SageMaker K-means clustering model.
- Train a custom XGBoost model.

Explanation:

B. Train an Amazon SageMaker DeepAR model.

DeepAR is a powerful time-series forecasting model that can handle complex patterns and generalize across multiple related time-series datasets. Since the company is introducing a new product variation, DeepAR can leverage historical sales data from similar products to generate accurate forecasts, even for items with little or no historical data.

Question: 369

Exam Heist

A Machine Learning Specialist must build out a process to query a dataset on Amazon S3 using Amazon Athena. The dataset contains more than 800,000 records stored as plaintext CSV files. Each record contains 200 columns and is approximately 1.5 MB in size. Most queries will span 5 to 10 columns only.

How should the Machine Learning Specialist transform the dataset to minimize query runtime?

- Convert the records to Apache Parquet format. ✓
- Convert the records to JSON format.
- Convert the records to GZIP CSV format.
- Convert the records to XML format.

Explanation:

The correct answer is **A. Convert the records to Apache Parquet format.**

Here's why:

Athena is designed to work efficiently with columnar data formats like Parquet. Columnar storage organizes data by columns rather than rows. This is crucial because the problem states that most queries only need 5-10 columns out of the 200 available. With a row-oriented format like CSV, Athena would have to scan the *entire* row for each record, even if only a few columns are needed, leading to wasted I/O and longer query times.

Parquet, on the other hand, allows Athena to read only the columns that are specified in the query. This drastically reduces the amount of data scanned, leading to significantly faster query execution and lower costs (since Athena charges based on data scanned). The compression capabilities of Parquet further reduce storage costs and the amount of data transferred during queries.

While converting to JSON, GZIP CSV, or XML might offer some level of compression compared to plaintext CSV, they are still fundamentally row-oriented formats. Therefore, they would not provide the same performance benefits as Parquet for analytical workloads like those performed with Athena where a subset of columns are frequently accessed. GZIP CSV offers compression but still requires reading entire rows. JSON and XML add significant overhead due to their verbose nature and are not optimized for analytical queries in Athena.
In summary, leveraging the columnar nature and compression of Parquet is the optimal strategy for minimizing Athena query runtime on this dataset by reducing I/O and focusing only on relevant columns.
Authoritative Links:

Amazon Athena Best Practices: https://docs.aws.amazon.com/athena/latest/ug/best-practices.html (specifically look for sections on file formats and columnar storage)
Apache Parquet: https://parquet.apache.org/ (official website for Parquet providing technical details)
AWS Big Data Blog - Top Performance Tuning Tips for Amazon Athena: https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/