1. **To label machine performance data using a simple heuristic based on the z-score, follow these steps:**

Step 1: Calculate the Z-Score

The z-score indicates how many standard deviations an element is from the mean. For each data point xix_ixi in your machine performance data, the z-score can be calculated as:

zi=xi−μσz_i = \frac{x_i - \mu}{\sigma}zi=σxi−μ

Where:

- xix_ixi is the data point (e.g., a machine's performance metric).
- μ\muμ is the mean of the performance data.
- σ\sigmaσ is the standard deviation of the performance data.

Step 2: Define Heuristic Thresholds

Define thresholds to label the data based on the z-score:

- **Normal Performance**: If ∣zi∣≤T|z_i| \leq T∣zi∣≤T (where TTT is a threshold, commonly set to 1 or 2).
- **Underperforming**: If zi<−Tz_i < -Tzi<−T.
- **Overperforming**: If zi>Tz_i > Tzi>T.

Step 3: Apply Labels

- **Label 1**: Normal performance if ∣zi∣≤T|z_i| \leq T∣zi∣≤T.
- **Label 0**: Underperforming if zi<−Tz_i < -Tzi<−T.
- **Label 2**: Overperforming if zi>Tz_i > Tzi>T.

Example:

Let's say you have a dataset of a machine's performance over time, measured as a numerical score.

1. Calculate the mean (μ\muμ) and standard deviation (σ\sigmaσ) of the performance data.
2. Calculate the z-score for each data point.
3. Apply the thresholds to label the data:
   - Normal (Label 1) if the z-score is between -1 and 1.
   - Underperforming (Label 0) if the z-score is less than -1.
   - Overperforming (Label 2) if the z-score is greater than 1.

Interpretation:

- **Label 1** indicates normal performance.
- **Label 0** indicates the machine is underperforming.
- **Label 2** indicates the machine is overperforming.

This heuristic approach can quickly give you a labeled dataset based on the machine's historical performance relative to its mean and variability.

2. The most appropriate technique to use in this situation is **L1 regularization** (also known as Lasso regularization) to reduce the coefficients of uninformative features to 0.

Why L1 Regularization?

- **Feature Selection**: L1 regularization is particularly effective in performing feature selection because it tends to shrink the coefficients of less informative features to exactly zero, effectively removing them from the model. This helps in identifying and retaining only the most informative features while removing the non-informative ones.
- **Simplicity and Interpretability**: By directly modifying the linear model, L1 regularization allows you to keep the informative features in their original form, which is important for interpretability.

L1 regularization is the most straightforward and effective method for feature selection in a linear model when dealing with a large number of input features, helping you retain only the most informative ones while eliminating the rest.

L1 regularization is a powerful tool in machine learning, particularly useful in certain scenarios. Here are the key situations where you should consider using L1 regularization:

1. Feature Selection

- **Sparse Solutions**: L1 regularization encourages sparsity in the model by shrinking the coefficients of less important features to exactly zero. This makes it effective for automatic feature selection, helping you identify the most relevant features while eliminating the non-informative ones.
- **High-Dimensional Data**: When you have a large number of features, especially more features than observations (e.g., in text classification or gene expression data), L1 regularization can help reduce the feature space by keeping only the most significant features.

2. Overfitting Prevention

- **Overfitting Control**: L1 regularization helps prevent overfitting by adding a penalty to the loss function based on the absolute values of the coefficients. This regularization term discourages the model from fitting the noise in the data, leading to a more generalized model.

3. When Feature Interpretability is Important

- **Model Interpretability**: Since L1 regularization can set some coefficients to zero, it produces a simpler, more interpretable model. This is useful when you want to understand which features are contributing to the predictions and to what extent.

4. When You Expect Only a Few Features to Be Important

- **Sparse Feature Sets**: If you believe that only a small subset of your features are actually relevant for predicting the outcome, L1 regularization can help by shrinking the coefficients of irrelevant features to zero, leaving only the important ones.

5. When Using Linear Models

- **Linear and Logistic Regression**: L1 regularization is commonly used in linear models (e.g., Lasso regression) and logistic regression when you want to enforce sparsity in the model. It is especially useful in scenarios where feature selection is as important as predictive accuracy.

6. When Dealing with Multicollinearity

- **Multicollinearity**: In situations where there is high multicollinearity (i.e., when features are highly correlated), L1 regularization can help by selecting one among the correlated features, effectively reducing redundancy and simplifying the model.

Summary:

Use L1 regularization when you need feature selection, want to prevent overfitting, require an interpretable model, or are dealing with high-dimensional data where only a few features are likely to be important. It is particularly well-suited for linear models where sparsity and interpretability are key concerns.

3. Data leakage in machine learning occurs when information from outside the training dataset is used to create the model, leading to overly optimistic performance estimates during model development but poor generalization to new data. This leakage results in a model that performs well during training or validation but fails in real-world scenarios because it inadvertently "learns" from data it shouldn't have access to.

Types of Data Leakage

1. **Target Leakage**: Happens when the model is trained on data that includes the target variable or information that directly correlates with the target in a way that would not be available in a real-world scenario.
2. **Train-Test Contamination**: Occurs when the training data is not properly separated from the test data, causing the model to learn from the test set.

Examples of Data Leakage

Example 1: Target Leakage

Suppose you're building a model to predict whether a customer will default on a loan. You include features like:

- Customer's income
- Loan amount
- Credit score
- **Repayment status**

Here, **repayment status** is a feature that indicates whether the loan has already been repaid. If this feature is included during training, the model will learn that loans marked as "repaid" are not defaults, which is trivial information that wouldn't be available at the time of prediction. This is target leakage because the model is trained on information that directly reveals the target outcome.

Example 2: Train-Test Contamination

Imagine you’re working on a time series problem where you want to predict stock prices. If you accidentally shuffle the data before splitting it into training and testing sets, some future data points might leak into the training set. As a result, the model might perform exceptionally well on the test data because it has already seen future values during training, leading to unrealistic performance estimates.

Consequences of Data Leakage

- **Overfitting**: The model learns patterns that are too closely tied to the specific data it was trained on, rather than general patterns.
- **Poor Real-World Performance**: Once deployed, the model's performance drops because it can’t rely on leaked information in real-world scenarios.
- **Misleading Metrics**: Validation or test metrics might indicate the model is highly accurate when, in reality, it’s performing poorly.

How to Prevent Data Leakage

- **Careful Feature Selection**: Ensure that features used during training are available at the time of prediction and don’t contain information that directly correlates with the target in a non-causal way.
- **Proper Data Splitting**: When splitting data into training and test sets, avoid shuffling time-series data or data where the order matters.
- **Cross-Validation**: Use techniques like k-fold cross-validation that help ensure the model is evaluated on unseen data.

Summary

Data leakage can severely compromise the integrity of your machine learning model by introducing information that the model wouldn't have access to in a real-world scenario. Being vigilant about the data used during training and validation is crucial to avoid this common pitfall.

Dynamic Range Quantization (DRQ) is a technique used in machine learning and neural networks to reduce the precision of the numerical values in a model, particularly focusing on quantizing the weights and activations to lower-bit representations. This process helps in reducing the model size and computational requirements, making it more efficient for deployment on resource-constrained devices like mobile phones and embedded systems.

Key Concepts of Dynamic Range Quantization

1. **Quantization**:
   - **Definition**: Quantization is the process of mapping a large set of values (e.g., floating-point numbers) to a smaller set of values (e.g., integer values). In the context of neural networks, this often involves converting weights and activations from 32-bit floating-point numbers to lower-bit integers (e.g., 8-bit integers).
   - **Purpose**: It reduces memory usage and accelerates inference by using less computational resources and bandwidth.
2. **Dynamic Range**:
   - **Definition**: Dynamic range refers to the range of values that a signal can take. In quantization, it specifically relates to the range of values for weights and activations that the quantization process needs to handle.
   - **Dynamic Range Quantization**: This technique involves scaling and quantizing weights and activations dynamically based on their actual distribution in the data, as opposed to static quantization, which uses fixed ranges for all values.
3. **Dynamic Range Quantization Process**:
   - **Scaling**: For each tensor (weights or activations), compute the maximum and minimum values. Scale these values to fit into the desired quantization range (e.g., from -127 to 127 for 8-bit signed integers).
   - **Quantizing**: Map the scaled values to discrete integer values within the quantization range.
   - **De-quantizing**: Convert the quantized values back to floating-point values for further computations during inference.

Benefits of Dynamic Range Quantization

- **Reduced Model Size**: By quantizing weights and activations, the model size decreases significantly, which is crucial for deploying models on devices with limited storage.
- **Faster Inference**: Lower precision computations are faster and require less energy, leading to faster inference times and improved efficiency.
- **Lower Power Consumption**: Reduced computational complexity translates to lower power usage, which is important for battery-operated devices.

Trade-offs and Considerations

- **Accuracy Loss**: Quantization can introduce a loss in model accuracy due to reduced precision. However, dynamic range quantization aims to minimize this loss by scaling and mapping values based on their actual distribution.
- **Calibration**: Proper calibration of the quantization process is essential to retain as much model accuracy as possible. This involves analyzing the distribution of weights and activations and choosing appropriate scaling factors.

AutoML Tables is a feature provided by Google Cloud's AutoML suite designed to automate the process of creating machine learning models for structured data (i.e., tabular data). It simplifies the model development process by automating key tasks, including data preprocessing, model selection, training, and hyperparameter tuning. Here's a detailed overview:

Key Features of AutoML Tables

1. **Automated Data Preparation**:
    - **Feature Engineering**: Automatically identifies and creates features from raw data. It performs operations like normalization, encoding categorical variables, and generating new features.
    - **Data Cleaning**: Handles missing values and performs other data cleaning tasks to prepare the dataset for modeling.
2. **Model Training and Selection**:
    - **Algorithm Selection**: Automatically selects and tunes various machine learning algorithms to find the best model for your data.
    - **Hyperparameter Tuning**: Optimizes hyperparameters for the selected models to improve performance.
3. **Performance Evaluation**:
    - **Model Evaluation**: Provides performance metrics and visualizations to help you understand how well the model is performing.
    - **Feature Importance**: Offers insights into which features are most important for model predictions.
4. **Ease of Use**:
    - **User-Friendly Interface**: Allows users to build models through a web-based interface without requiring extensive machine learning expertise.

    - **Integration**: Seamlessly integrates with other Google Cloud services for data storage, visualization, and deployment.
5. **Deployment and Monitoring**:
    - **Model Deployment**: Facilitates easy deployment of trained models to production environments.
    - **Monitoring**: Provides tools for monitoring model performance over time and retraining models as needed.

Typical Use Cases

- **Predictive Analytics**: Used for tasks such as forecasting sales, predicting customer churn, and estimating financial metrics.
- **Classification and Regression**: Handles various types of tabular data problems, including classification (e.g., categorizing customer segments) and regression (e.g., predicting house prices).
- **Business Intelligence**: Helps businesses leverage their data for insights and decision-making without needing specialized data science skills.

Example Workflow

1. **Upload Data**: Start by uploading your tabular dataset to Google Cloud Storage or directly into AutoML Tables.
2. **Configure the Dataset**: Specify the target column (the variable you want to predict) and configure any additional settings.
3. **Train the Model**: AutoML Tables automatically preprocesses the data, selects algorithms, trains models, and tunes hyperparameters.
4. **Evaluate**: Review the performance metrics and feature importance provided by AutoML Tables.
5. **Deploy**: Deploy the model to a production environment and use it for making predictions.
6. **Monitor**: Keep track of the model's performance and retrain it as necessary based on new data or changing conditions.

Summary

AutoML Tables is a powerful tool for automating the creation and deployment of machine learning models for structured data. It streamlines the process by handling data preparation, model training, hyperparameter tuning, and evaluation, making it accessible even to those without extensive machine learning expertise.

Dialogflow is a natural language understanding platform that makes it easy to design and integrate a conversational user interface into your mobile app, web application, device, bot, interactive voice response system, and so on. Using Dialogflow, you can provide new and engaging ways for users to interact with your product.

Local feature importance gives you visibility into how the individual features in a specific prediction request affected the resulting prediction. Each local feature importance value shows only how much the feature affected the prediction for that row. To understand the overall behavior of the model, use model feature importance.“

The **AutoML Vision Edge mobile-low-latency-1** model is part of Google’s AutoML Vision Edge suite, specifically optimized for scenarios requiring low-latency image classification on mobile and edge devices. Here’s a detailed overview:

Key Features

1. **Low Latency**:
    - Designed for real-time image classification tasks where quick response times are crucial. This model aims to minimize the delay between image capture and classification output.
2. **Mobile Optimization**:
    - Tailored for deployment on mobile devices and edge hardware, ensuring that it operates efficiently within the constraints of these environments.
3. **Pre-trained Model**:
    - Comes pre-trained on a broad set of image data, providing robust performance across various image classification tasks. You can fine-tune it with your own data to meet specific requirements.
4. **Efficient Computation**:
    - The model is optimized for minimal computational resources, balancing performance and resource usage, which is ideal for devices with limited processing power.

Use Cases

1. **Real-Time Applications**:
    - Suitable for applications where immediate feedback is essential, such as augmented reality, real-time object detection, and interactive applications.
2. **Mobile Devices**:
    - Ideal for integration into mobile apps where low-latency responses are necessary for user experience, such as in-camera apps or on-device image analysis.
3. **Edge Devices**:

    - Useful in embedded systems where processing power and memory are constrained but quick classification is needed, such as in smart cameras or IoT devices.

Deployment Steps

1. **Training or Fine-Tuning**:
    - Utilize AutoML Vision Edge to train or fine-tune the mobile-low-latency-1 model with your dataset. This helps customize the model to your specific application and data characteristics.
2. **Exporting the Model**:
    - Export the trained or fine-tuned model in TensorFlow Lite (TFLite) format, which is optimized for mobile and edge deployments. This format ensures that the model can be efficiently loaded and run on the target devices.
3. **Integration into Mobile or Edge Devices**:
    - Integrate the TFLite model into your mobile app or edge device using TensorFlow Lite. This involves loading the model, performing inference, and processing the results.

—-----------------------------

Categorical Cross-Entropy and **Sparse Categorical Cross-Entropy** are both loss functions used in classification tasks involving multiple classes. They measure the difference between the true labels and the predicted probabilities. The key difference between them lies in how the true labels are represented and processed:

1. Categorical Cross-Entropy

Label Representation: The true labels are **one-hot encoded**. This means that for each sample, the true label is represented as a vector where the correct class is marked as `1`, and all other classes are marked as `0`.
For example, if there are three classes, a label might be represented as:
csharp
Copy code

```
[1, 0, 0]  # Class 1
[0, 1, 0]  # Class 2
[0, 0, 1]  # Class 3
```

-
- **Use Case**: Use Categorical Cross-Entropy when your labels are already in one-hot encoded form or when you manually handle the one-hot encoding.

- **Formula**:
  Loss=−∑i=1Cyi⋅log(y^i)\text{Loss} = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i)Loss=−i=1∑Cyi⋅log(y^i)
  Where:
  - CCC is the number of classes.
  - yiy_iyi is the true label (one-hot encoded).
  - y^i\hat{y}_iy^i is the predicted probability for class iii.

2. Sparse Categorical Cross-Entropy

Label Representation: The true labels are **integers** corresponding to the correct class index. There is no need for one-hot encoding.
For example, if there are three classes, a label might be represented as:
bash
Copy code

```
0  # Class 1
1  # Class 2
2  # Class 3
```

-
- **Use Case**: Use Sparse Categorical Cross-Entropy when your labels are integers representing the class index. This is often more memory-efficient, especially when dealing with a large number of classes.
- **Formula**:
  Loss=−log(y^true)\text{Loss} = -\log(\hat{y}_{\text{true}}) Loss=−log(y^true)
  Where:
  - y^true\hat{y}_{\text{true}}y^true is the predicted probability for the true class label.

Summary

- **Categorical Cross-Entropy**: Requires one-hot encoded labels.
- **Sparse Categorical Cross-Entropy**: Requires integer labels representing class indices.

Practical Tip: Use Sparse Categorical Cross-Entropy when you have a large number of classes or when you prefer not to one-hot encode your labels, as it can be more efficient in terms of memory usage.

4o

Qestions

Working for an ad company, one needs to comprehend the efficiency of the recent advertising campaign. By streaming 500MB of campaign data into BigQuery, one can query the table, and then manipulate the results of the query utilizing a pandas dataframe in an AI Platform notebook. What is the necessary action to achieve this?

Working for an ad company, one needs to comprehend the efficiency of the recent advertising campaign. By streaming 500MB of campaign data into BigQuery, one can query the table, and then manipulate the results of the query utilizing a pandas dataframe in an AI Platform notebook. What is the necessary action to achieve this?

A .Download your table from BigQuery as a local CSV file, and upload it to your AI Platform notebook instance. Use pandas.read_csv to ingest the file as a pandas dataframe.

B. Export your table as a CSV file from BigQuery to Google Drive, and use the Google Drive API to upload the file into your notebook instance.

C . From a bash cell in your AI Platform notebook, use the bq extract command to export the table as a CSV file to Cloud Storage, and then use gsutil cp to transfer the data into the notebook. Use pandas.read_csv to ingest the file as a pandas dataframe.

D. Use AI Platform Notebooks‘ BigQuery cell magic to query the data and ingest the results as a pandas dataframe.

Ans : D

Training an object detection machine learning model on a dataset of three million X-ray images, each approximately 2 GB in size, can be an arduous task. Utilizing Vertex AI Training to run a custom training application on a Compute Engine instance with 32 cores, 128 GB of RAM, and 1 NVIDIA P100 GPU, you have observed that the model training is taking an extended period of time. To decrease training time without compromising model performance, what are the potential solutions?

Utilize the tf.distribute.Strategy API and execute a distributed training job.

Substitute the NVIDIA P100 GPU with a v3-32 TPU in the training job.

Include early stopping in your Vertex AI Training job.

Increase the instance memory to 512 GB and the batch size.

Ans - C

In order to create an ML model to accurately classify X-ray images for bone fracture risk, you have used the ResNet architecture on Vertex AI with a TPU as an accelerator. Despite the training time and memory usage, you are not satisfied with the results. To quickly iterate the training code while minimizing changes and impact on accuracy, what would be the best solution?

- Decrease the dimensions of the images used in the model.
- Set up your model to use bfloat16 instead of float32.
- Decrease the number of layers in the model architecture.
- Decrease the global batch size from 1024 to 256.

Ans - B

You need to design an architecture that serves asynchronous predictions to determine whether a particular mission-critical machine part will fail. Your system collects data from multiple sensors from the machine. You want to build a model that will predict a failure in the next N minutes, given the average of each sensor' s data from the past 12 hours. How should you design the architecture?

- 1. Export the data to Cloud Storage using the BigQuery command-line tool
  2. Submit a Vertex AI batch prediction job that uses your trained model in Cloud Storage to perform scoring on the preprocessed data.
  3. Export the batch prediction job outputs from Cloud Storage and import them into BigQuery.
- 1. Events are sent by the sensors to Pub/Sub, consumed in real time, and processed by a Dataflow stream processing pipeline.
  2. The pipeline invokes the model for prediction and sends the predictions to another Pub/Sub topic.
  3. Pub/Sub messages containing predictions are then consumed by a downstream system for monitoring.
- **1. Export your data to Cloud Storage using Dataflow.**
  2. Submit a Vertex AI batch prediction job that uses your trained model in Cloud Storage to perform scoring on the preprocessed data.
  3. Export the batch prediction job outputs from Cloud Storage and import them into Cloud SQL.
- 1. HTTP requests are sent by the sensors to your ML model, which is deployed as a microservice and exposes a REST API for prediction

2. Your application queries a Vertex AI endpoint where you deployed your model.
3. Responses are received by the caller application as soon as the model produces the prediction.

Ans 3(Pub/sub)

<< **1. Events are sent by the sensors to Pub/Sub, consumed in real time, and processed by a Dataflow stream processing pipeline. 2. The pipeline invokes the model for prediction and sends the predictions to another Pub/Sub topic. 3. Pub/Sub messages containing predictions are then consumed by a downstream system for monitoring.**
>>

As an operations team at an international corporation, your task is to manage a considerable number of on-premises servers located in few data centers worldwide. From the servers, your team collects CPU/memory usage data. When any incident occurs, your team is required to take action. Unfortunately, the incident data has still not been properly labeled. Your management team has asked you to construct a predictive maintenance solution that can use monitoring data from the VMs to recognize probable failures and then alert the service desk team. So, what would be the initial step?

Create a straightforward heuristic (e.g., based on z-score) to classify the machines' historical performance data. Assess this heuristic in a production environment.

Design a straightforward heuristic (e.g., based on z-score) to classify the machines' historical performance data. Construct a model to recognize deviations based on this classified dataset.

Train a time-series model to forecast the machines' performance values. Establish an alert if a machine's actual performance values differ significantly from the predicted performance values.

Recruit a team of skilled analysts to examine and classify the machines' historical performance data. Construct a model based on this manually classified dataset.

Ans A

Your team is constructing an application for a multinational bank that will be utilized by an immense number of customers. You developed a forecasting model that prognosticates customers' account balances three days in advance. Your team intends to apply the results in a novel feature that will advise users when their account balance has a high probability of falling beneath $25. How should you present your predictions?

1. Construct a notification system on Firebase. 2. Register each user with a user ID on the Firebase Cloud Messaging server, which sends a notification when the average of all account balance predictions drops below the $25 threshold.

1. Create a Pub/Sub topic for each user. 2. Deploy an application on the App Engine standard environment that sends a notification when your model predicts that a user‘s account balance will drop below the $25 threshold.

1. Construct a notification system on Firebase. 2. Register each user with a user ID on the Firebase Cloud Messaging server, which sends a notification when your model predicts that a user‘s account balance will drop below the $25 threshold.

1. Create a Pub/Sub topic for each user. 2. Deploy a Cloud Function that sends a notification when your model predicts that a user‘s account balance will drop below the $25 threshold.

Ans C

Your team is constructing a Convolutional Neural Network (CNN) based architecture from the start. Initial tests on your on-premises CPU-only infrastructure were promising, however, progression was slow. To decrease time-to-market, you have been requested to quicken model training. You plan to try out VMs on Google Cloud to benefit from stronger hardware. Your code has not been equipped with any manual device placement and is not contained in Estimator model-level abstraction. What environment should be used to train your model?

AVM on Compute Engine and 8 GPUs with all dependencies installed manually.

AVM on Compute Engine and 1 TPU with all dependencies installed manually.

A Deep Learning VM with an e2-highcpu-16 machine and all libraries pre-installed.

A Deep Learning VM with an n1-standard-2 machine and 1 GPU with all libraries pre-installed.

Ans - B

To create a tailored Deep Neural Network in Keras that predicts customer purchases based on their past transactions, it is necessary to investigate model performance using diverse model structures, store the training data and have the capacity to contrast evaluation metrics on the same dashboard. What is the best way to do this?

Execute multiple training jobs on AI Platform with similar job names.

Automate multiple training iterations using Cloud Composer.

- Build an experiment in Kubeflow Pipelines to organize multiple runs.
- Generate various models using AutoML Tables.

Ans A

After successfully training and testing a DNN regression model, it was deployed with successful results. However, after six months, the model's performance has deteriorated due to a change in the distribution of the input data. What is the best way to tackle the disparities between the input data in production?

- Retrain the model, and pick an L2 regularization parameter with a hyperparameter tuning service.
- Set up alarms to track skew, and retrain the model.
- Do feature selection on the model, and retrain the model using fewer features.
- Do feature selection on the model, and retrain the model on a monthly basis using fewer features.

Ans B

Your team has been tasked with creating a model that can accurately distinguish between images of driver's licenses, passports, and credit cards. The data engineering team has already compiled a dataset with 10,000 images of driver's licenses, 1,000 of passports, and 1,000 of credit cards, and assigned the labels ˜drivers_license', ˜passport', and ˜credit_card' respectively. With this in mind, what loss function should be used to train the model?

- Categorical hinge
- Categorical Cross Entropy
- Binary Cross Entropy
- Sparse Categorical Cross Entropy

ANS - B

To enhance the input/output performance of a TensorFlow model trained on a structured dataset containing 100 billion records distributed among various CSV files, what steps can be taken?

- Convert the CSV files into shards of TFRecords, and store the data in Cloud Storage.
- Convert the CSV files into shards of TFRecords, and store the data in the Hadoop Distributed File System (HDFS).
- Load the data into Cloud Bigtable, and read the data from Cloud Bigtable.
- Load the data into BigQuery, and read the data from BigQuery

Ans - A

To build classification workflows over numerous datasets stored in BigQuery, without writing code, it is necessary to complete steps such as exploratory data analysis, feature selection, model building, training, and hyperparameter tuning and serving. In order to repeat the classification multiple times, what should be done?

- Utilize AI Platform to run the classification model job tailored for hyperparameter optimization.
- Employ AI Platform Notebooks to run the classification model with pandas library.
- Configure AutoML Tables to perform the classification task.
- Execute a BigQuery ML job to perform logistic regression for the classification.

Ans - C

Hypertuning your ML model with AI Platform is taking longer than expected and delaying the downstream processes of your end-to-end ML pipeline. To speed up the process without significantly reducing its effectiveness, what steps should you take? (Choose two.)

- Reduce the range of floating-point values.
- Set the early stopping parameter to TRUE.
- Lower the maximum number of trials during subsequent training phases.
- Substitute the Bayesian search with random search.
- Decrease the number of parallel trials.

Ans - A,D

As a Machine Learning Engineer at a regulated insurance company, it is important to consider a few factors before building a model to approve or reject insurance applications from potential customers. These factors should be taken into account to create an effective and compliant model. What are these considerations?

- Redaction, reproducibility, and interpretability
- Federated learning, reproducibility, and interpretability
- Differential privacy, federated learning, and interpretability
- Traceability, reproducibility, and interpretability

Ans C

To optimize the efficiency of their data science team, they must quickly test different features, model structures, and hyperparameters. To accurately monitor the results of these experiments, they require an API to access metrics over time. What tool could they use to accurately track and report their experiments while minimizing manual labor?

- Use AI Platform Training to execute the experiments. Record the accuracy metrics in BigQuery, and query the results via the BigQuery API.
- Utilize AI Platform Notebooks to run the experiments. Store the results in a shared Google Sheets file, and query the results via the Google Sheets API.
- Use AI Platform Training to execute the experiments. Record the accuracy metrics in Cloud Monitoring, and query the results via the Monitoring API.
- Utilize Kubeflow Pipelines to run the experiments. Export the metrics file, and query the results via the Kubeflow Pipelines API.

Ans - D

During an investigation into a dataset, it is discovered that categorical feature A has considerable influence in prediction, however, it is occasionally absent. What is the best course of action in this situation?

- Drop feature A if more than 15% of values are missing. Otherwise, use feature A as-is.
- Replace the missing values with the values of the feature with the highest Pearson correlation with feature A.
- Add an additional class to categorical feature A for missing values. Create a new binary feature that indicates whether feature A is missing.
- Compute the mode of feature A and then use it to replace the missing values in feature A.

Ans - C

After preprocessing data stored in Parquet files and accessed through a Hive table hosted on Google Cloud with PySpark, and exporting it as a CSV file to Cloud Storage, you have built a model for training and evaluation. To parametrize this model training, what should you do next with Kubeflow Pipelines?

- Set up Apache Spark in a distinct node pool in a Google Kubernetes Engine cluster. Put a ContainerOp in your pipeline that starts a linked transformation job for this Spark instance.
- Put the PySpark transformation step in a container, and incorporate it to your pipeline.
- Put a ContainerOp in your pipeline that turns on a Dataproc cluster, does a transformation, and then deposits the changed data in Cloud Storage.
- Take out the data transformation step from your pipeline.

ANS - C

As part of an AI team for an automobile company, you are creating a visual defect detection model with TensorFlow and Keras. To enhance its performance, you are randomly incorporating image augmentation functions such as translation, cropping and contrast tweaking for each training batch. To optimize data processing pipeline for run time and compute resources utilization, what measures can you take?

- Insert the augmentation functions dynamically in the tf.Data pipeline.
- Utilize Dataflow to create the augmentations dynamically for each training run, and store them as TFRecords.
- Utilize Dataflow to generate all possible augmentations, and save them as TFRecords.
- Insert the augmentation functions dynamically as part of Keras generators.

ANS - A

To quickly scale your training workload while minimizing cost, you are developing an image recognition model using PyTorch based on the ResNet50 architecture. Your code is functioning properly on a small subsample of your full dataset, which comprises 200k labeled images. To achieve this, you intend to leverage the power of four V100 GPUs. What steps should you take to make this a reality?

- Create a Google Kubernetes Engine cluster with a node pool that has 4 V100 GPUs. Prepare and submit a TFJob operator to this node pool.
- Generate a Vertex AI Workbench user-managed notebooks instance with 4 V100 GPUs, and use it to train your model.
- Set up a Compute Engine VM with all the necessary dependencies that launches the training. Train your model with Vertex AI using a custom tier that contains the required GPUs.
- Package your code with Setuptools, and utilize a pre-built container. Train your model with Vertex AI using a custom tier that contains the required GPUs.

ANS - C

As a Machine Learning Engineer at a social media company, I am developing a visual filter for users‘ profile photos. This necessitates training an ML model to recognize bounding boxes around human faces. In order to reduce code development and optimize the model for inference on mobile phones, what should I do to achieve this goal for my company‘s iOS-based mobile phone application?

- Train a custom TensorFlow model and convert it to TensorFlow Lite (TFLite).
- Train a model using AutoML Vision and use the “export for TensorFlow.js” option.
- Train a model using AutoML Vision and use the “export for Coral” option.
- Train a model using AutoML Vision and use the “export for Core ML” option.

ANS - A

You work for a company that manages a ticketing platform for a large chain of cinemas. Customers use a mobile app to search for movies they’re interested in and purchase tickets in the app. Ticket purchase requests are sent to Pub/Sub and are processed with a Dataflow streaming pipeline configured to conduct the following steps:
1. Check for availability of the movie tickets at the selected cinema.
2. Assign the ticket price and accept payment.
3. Reserve the tickets at the selected cinema.
4. Send successful purchases to your database.

Each step in this process has low latency requirements (less than 50 milliseconds). You have developed a logistic regression model with BigQuery ML that predicts whether offering a promo code for free popcorn increases the chance of a ticket purchase, and this prediction should be added to the ticket purchase process. You want to identify the simplest way to deploy this model to production while adding minimal latency. What should you do?

- 🟩 **Export your model in TensorFlow format, deploy it on Vertex AI, and query the prediction endpoint from your streaming pipeline.**
- 🟥 **Convert your model with TensorFlow Lite (TFLite), and add it to the mobile app so that the promo code and the incoming request arrive together in Pub/Sub.**
- 🟥 Run batch inference with BigQuery ML every five minutes on each new set of tickets issued.
- 🟥 Export your model in TensorFlow format, and add a tfx_bsl.public.beam.RunInference step to the Dataflow pipeline.

Ans - A

Having conducted a few experiments using random cross-validation on a classification problem with time series data, a high Area Under the Receiver Operating Characteristic Curve (AUC ROC) of 99% was achieved on the training data, yet no sophisticated algorithms or hyperparameter tuning had been explored. What should be the next step to identify and address the issue?

- 🟩 Combat model overfitting by adjusting the hyperparameters to decrease the AUC ROC value.
- 🟩 Combat data leakage by eradicating features highly correlated with the target value.
- 🟩 Combat model overfitting by utilizing a less complex algorithm and k-fold cross-validation.
- 🟥 Combat data leakage by utilizing nested cross-validation during model training.

Ans -D

After deploying a complex TensorFlow model trained on tabular data to production, you want to predict the lifetime value (LTV) field for each subscription stored in the BigQuery table named ‘subscriptionPurchase‘ in the project named ‘my-fortune500-company-project‘. To make this process more efficient, you have organized all your training code from preprocessing data up to deploying the validated model to the Vertex AI endpoint into a TensorFlow Extended (TFX) pipeline. To prevent prediction drift, i.e. a situation when a feature data distribution changes significantly over time, what should you do?

Add a model monitoring job where 90% of incoming predictions are sampled every 24 hours.

Add a model monitoring job where 10% of incoming predictions are sampled every hour.

Implement continuous retraining of the model daily using Vertex AI Pipelines.

Add a model monitoring job where 10% of incoming predictions are sampled every 24 hours.

Ans - B