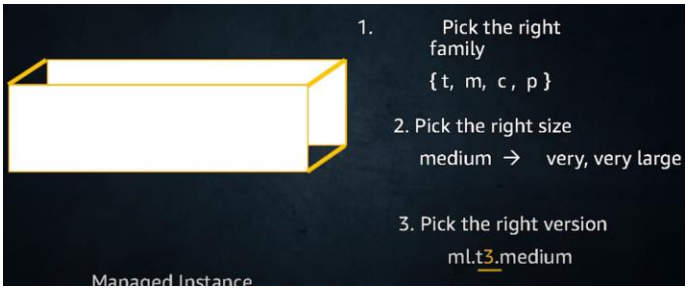
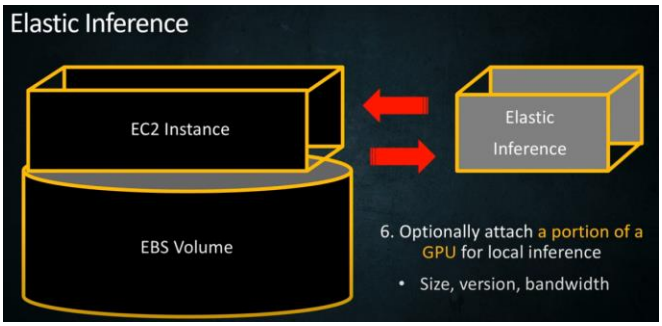
Domain X: Misc

X.1 SageMaker Deep Dive

X.1.1 Fully Managed Notebook Instances with Amazon SageMaker





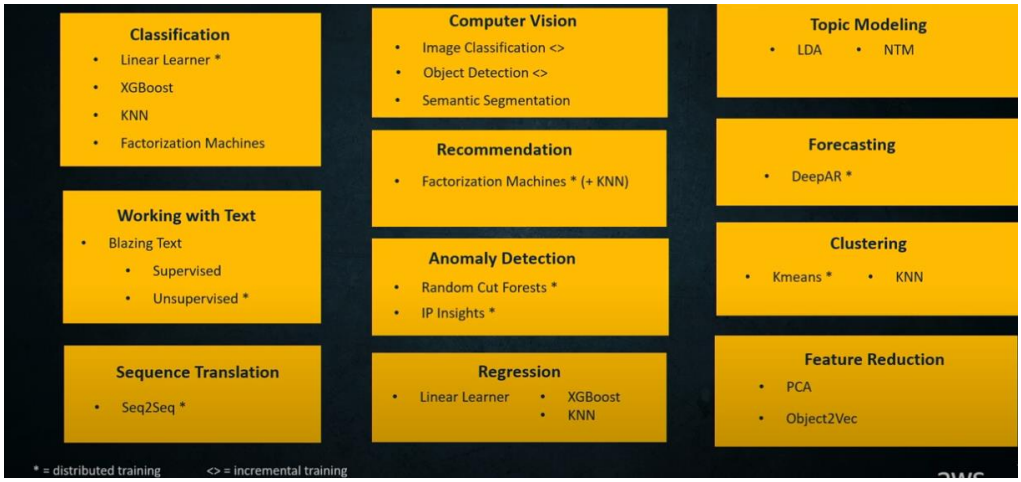
Elastic Inference

Elastic Inference is a service that allows attaching a portion of a GPU to an existing EC2 instance.2 This approach is particularly useful when running inference locally on a notebook instance.2 By selecting an appropriate Elastic Inference configuration based on size, version, and bandwidth, users can accelerate their inference tasks without needing a full GPU.2

Use Cases for Elastic Inference

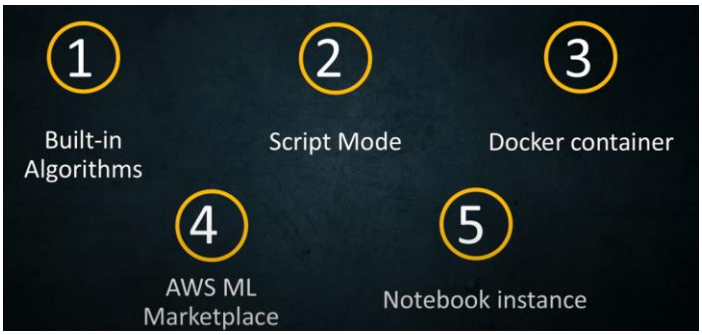
- You need to run inference tasks locally on your notebook instance.
- Your workload benefits from GPU acceleration but doesn't require a full GPU.
- You want to optimize cost by only paying for the portion of GPU resources used.

X.1.2 SageMaker Built-in Algorithms



Task Category	Algorithms	Supervised/UnSupervised
Classification	• Linear Learner (distributed) • XGBoost • KNN • Factorization Machines	Supervised
Regression	• Linear Learner • XGBoost • KNN	Supervised
Computer Vision	• Object Detection (incremental) • Semantic Segmentation	Supervised
Working with Text	• BlazingText	Supervised / Unsupervised
Sequence Translation	• Seq2Seq (distributed)	Supervised
Recommendation	• Factorization Machines (distributed) • KNN	Supervised
Anomaly Detection	• Random Cut Forests (distributed) • IP Insights (distributed)	Unsupervised / Semi-supervised
Topic Modeling	• LDA • NTM	Unsupervised
Forecasting	• DeepAR (distributed)	Supervised
Clustering	• K-means (distributed) • KNN	Unsupervised
Feature Reduction	• PCA • Object2Vec	Unsupervised / Semi-supervised

X.1.3 SageMaker Training types



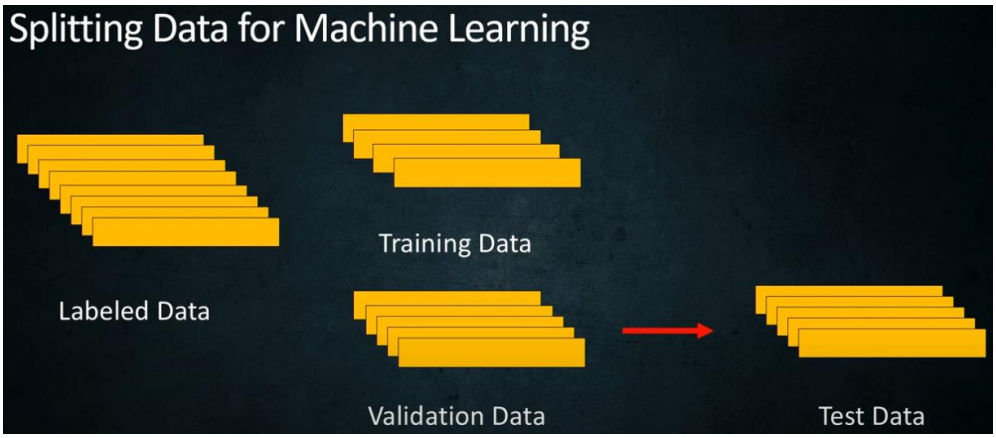
Training Type	Description	When to Use
1. Built-in Algorithms	Pre-configured algorithms provided by Amazon SageMaker, optimized for performance and ease of use	• Working on common ML tasks (e.g., classification, regression) • When you need a quick start without deep ML expertise
2. Script Mode	Custom training scripts using popular ML frameworks (e.g., TensorFlow, PyTorch, Scikit-learn)	• You have existing scripts in popular ML frameworks • For customizing model architecture while leveraging SageMaker's infrastructure
3. Docker Container	Custom Docker containers with your own algorithms or environments	• Need complete control over training env. • Custom or proprietary algorithms • For complex, multi-step training pipelines
4. AWS ML Marketplace	Pre-built algorithms and models from third-party vendors available through the AWS Marketplace	• Need industry-specific or specialized models • When you want to explore alternative solutions without building from scratch
5. Notebook Instance	Interactive development and training using Jupyter notebooks on managed instances	• During the initial stages of model development • When you need an interactive environment for debugging and visualization

Key Considerations:

- Skill Level: Built-in Algorithms and Marketplace for beginners, Script Mode and Containers for more advanced users
- Customization Needs: From low (Built-in) to high (Containers)
- Development Speed: Notebooks for rapid prototyping, Built-in for quick deployment, Containers for complex but reproducible setups
- Scale: Consider moving from Notebooks to other options as your data and model complexity grow.

X.1.4 Train Your ML Models with Amazon SageMaker

Splitting Data for ML



Create Notebook Instance → **Prepare Data** → Train & Validate the Model → Build the Model → Deploy the Model

Applying dummies as one-hot encoding to convert categorical features to numeric.

```
model_data = pd.get_dummies(churn)
model_data = pd.concat(
        [model_data["Churn?_True."],
        model_data.drop(
            ["Churn?_False.",
             "Churn?_True."],
             axis=1)],
        axis=1 )
```

Training data, and subsequent validation data, are stored in diff S3 buckets

```
s3_input_train = TrainingInput(
        s3_data=S3_train_location,
        content_type="csv" )
s3_input_validation = TrainingInput(
        s3_data= S3_validate_location,
        content_type="csv" )
```

IAM Role
# of instances
Type of instance

Training data, and subsequent validation data, are stored in diff S3 buckets

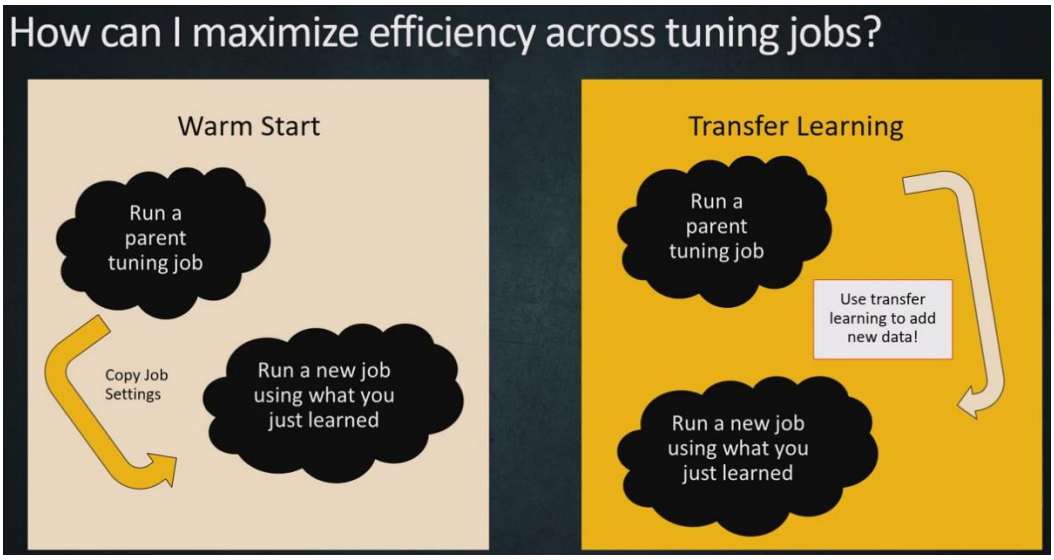
```
sess = sagemaker.Session()

xgb = sagemaker.estimator.Estimator
    (container,
     role,
     train_instance_count=1,
     train_instance_type='ml.m4.xlarge',
     output_path=s3_output_location,
     sagemaker_session=sess)

xgb.set_hyperparameters(
                max_depth=5,
                eta=0.2,
                gamma=4,
                min_child_weight=6,
                subsample=0.8,
                silent=0,
                objective='binary:logistic',
                num_round=100)

xgb.fit({
        'train': s3_input_train,
        'validation': s3_input_validation})
```

```
xgb_predict = xgb.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium')
xgb_predict.content_type = 'text/csv'
xgb_predict.serializer = csv_serializer
```

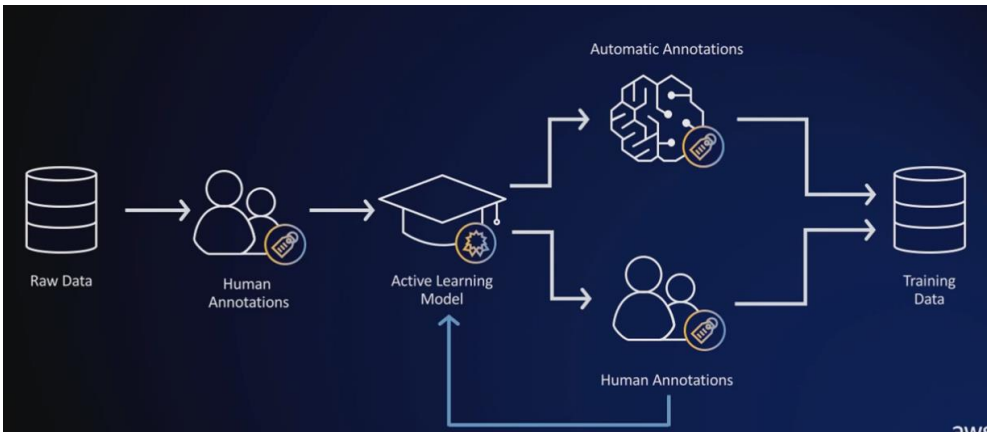
Deploy also involves creating an API endpoint

X.1.5 Tuning Your ML Models with Amazon SageMaker

Maximizing Efficiency across tuning jobs



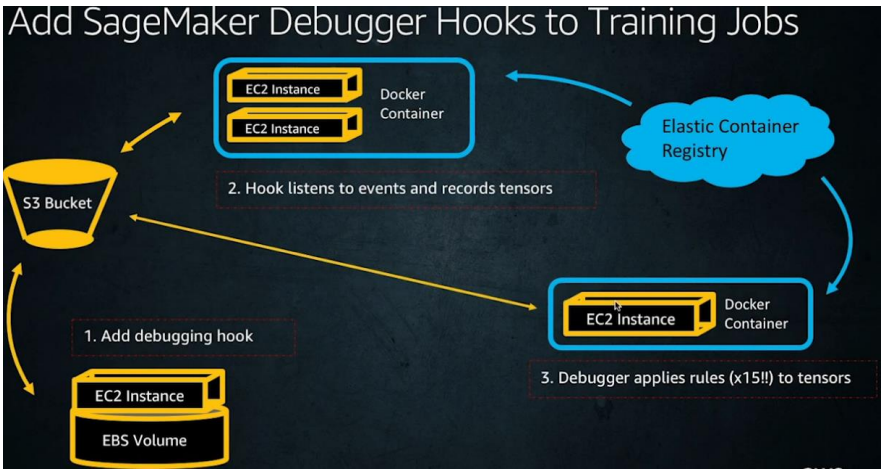
X.1.6 Tuning Your ML Models with Amazon SageMaker



How to automate

Put a check to see if Accuracy falls below a % (e.g. > 80%), invoke Human in the loop

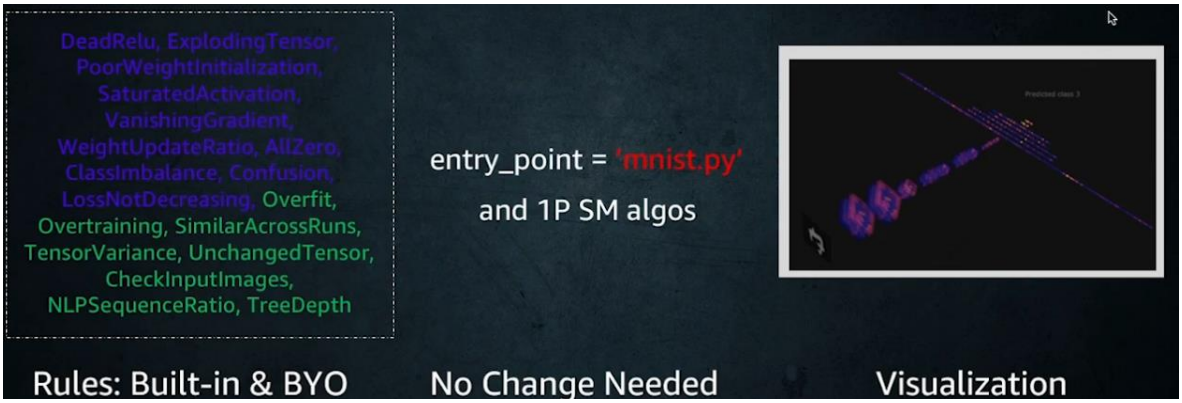
X.1.6 Add Debugger to Training Jobs in Amazon SageMaker



How it works

1. Add debugging hook:
    - An EC2 instance with an attached EBS volume is used to initiate the process.
    - The debugging hook is added to the training job configuration.
2. Hook listens to events and records tensors:
    - Docker containers running on EC2 instances are used for the training job.
    - The hook listens for specific events during the training process and records tensor data.
3. Debugger applies rules to tensors:
    - Another EC2 instance with a Docker container is used for debugging.
    - The debugger applies predefined rules (mentioned as "x15!!" in the image) to the recorded tensor data.

Benefits of debugger



1. Comprehensive Built-in Rules/Algorithms: The debugger offers a wide range of built-in rules to detect common issues in machine learning models, such as:
    - DeadRelu, ExplodingTensor, PoorWeightInitialization
    - SaturatedActivation, VanishingGradient
    - WeightUpdateRatio, AllZero, ClassImbalance
    - Confusion, LossNotDecreasing, Overfit
    - Overtraining, SimilarAcrossRuns
    - TensorVariance, UnchangedTensor
    - CheckInputImages, NLPSequenceRatio, TreeDepth

2. Customizable (BYO - Bring Your Own): Users can create and add their own custom debugging rules.
3. Easy Integration: The entry point is 'mnist.py' and it works with SageMaker's built-in algorithms (1P SM algos), suggesting easy integration with existing SageMaker workflows.
4. No Code Changes Required: The "No Change Needed" text implies that adding debugging capabilities doesn't require modifying the existing model code.
5. **Visualization**: The debugger provides visualization capabilities, as indicated by the image on the right, which appears to show a tensor or weight distribution.
6. **Real-time Monitoring**: The variety of rules suggests that the debugger can monitor various aspects of model training in real-time, helping to identify issues as they occur.

X.1.7 Deployment using SageMaker

Deployment Strategy	Description	When to Use
Blue/Green Deployment with Linear Traffic Shifting	Gradually shift traffic from the old version (blue) to the new version (green) over time	• When you need fine-grained control over the traffic shift • For critical applications requiring minimal risk • When you have the resources to run two full environments simultaneously
Canary Deployment	Release a new version to a small subset of users before rolling it out to the entire infrastructure	• When you want to test in production with real users • For early detection of issues before full deployment • When you have a diverse user base
A/B Testing	Run two versions simultaneously and compare their performance based on metrics	• When you want to test specific features or changes • When you need to optimize based on user behavior or business metrics
Rolling Deployment	Gradually replace instances of the old version with the new version	• When you have limited resources and can't run two full environments • For applications that can handle mixed versions • When you need to minimize downtime

X.2 From the Exam Guide

X.2.1 Domain 1: Data Preparation for Machine Learning (ML)

Data formats and ingestion mechanisms

	Format	Type	Description	Advantages	Common Use Cases
ROW BASED	CSV	Text	Simple tabular format	• Human-readable • Widely supported • Easy to generate	• Simple data exchange • Small to medium datasets
	JSON	• Semi-structured • Text	Flexible format	• Human-readable • Supports nested structures • Language-independent	• Web APIs • Configuration files • Document databases
	Apache Avro	Binary	Data serialization	• Compact serialization • Language-independent	• Data serialization • RPC protocols • Hadoop data storage
	RecordIO	Binary	SageMaker-specific format for efficient data loading	• Optimized for SageMaker • Supports large datasets	• SageMaker model training • Large-scale ML datasets
COMLUMNAR	Apache Parquet	Binary	Optimized format	• Efficient compression • Fast query performance • Schema evolution support	• Big data analytics • Data warehousing • Machine learning datasets
	Apache ORC (Optimized Row Columnar)	Binary	Optimized for Hadoop	• High compression ratio • Fast data processing	• Hive data storage • Big data processing • Analytics workloads

Core AWS data sources

Feature	S3	EFS	FSx for NetApp ONTAP
Best for	Large, infrequent-change data	Shared, frequent-change data	High-performance, multi-protocol
Latency	Higher	Low	Lowest
Scalability	Virtually unlimited	Up to petabytes	Up to hundreds of petabytes
Cost	Lowest	Moderate	Highest
ML Use Case	Training data, model artifacts	Distributed training, real-time	High-performance computing, Windows ML
Storage Type	Object storage	File storage	High-performance file storage
Access Pattern	Good for sequential access	Good for random access	Excellent for all access patterns
Shared Access	Not native	Native	Native
Protocols	S3 API, HTTP/S	NFS	NFS, SMB, iSCSI

Domain 2: ML Model Development

Common regularization techniques

Technique	Description	Benefits	Best Used When
Dropout	Randomly "drops out" a proportion of neurons during training	• Reduces overfitting • Improves generalization • Acts as an ensemble method • Prevents co-adaptation of features	• Large neural networks • Limited training data • Complex tasks with risk of overfitting
Weight Decay (L2)	Adds a penalty term to the loss function based on the squared magnitude of weights	• Prevents large weights • Improves generalization • Stabilizes learning • Helps with feature selection	• Most neural networks • When you want to keep all features but reduce their impact • When dealing with multicollinearity
L1 Regularization	Adds a penalty term to the loss function based on the absolute value of weights	• Encourages sparsity in the model • Feature selection (drives some weights to zero) • Robust to outliers • Computationally efficient for sparse data	• When feature selection is important • Dealing with high-dimensional data • When you want a sparse model

Open Source frameworks for SageMaker script mode - TensorFlow vs PyTorch

Feature	*TensorFlow*	*PyTorch*
Type	Open-source ML framework	Open-source ML framework
Specialization	General-purpose, excels in production deployment	Flexible, great for research and prototyping
Distributed Training	Supported via Horovod or parameter servers	Supported via PyTorch Distributed
~~***GPU Acceleration***~~	~~Fully supported~~	~~Fully supported~~
~~***Model Serving***~~	~~Native support in SageMaker~~	~~Native support in SageMaker~~
~~***Automatic Model Tuning***~~	~~Supported~~	~~Supported~~

Domain 4: ML Solution Monitoring, Maintenance, and Security

Design principles for ML lenses relevant to monitoring

Principle	Key Points
1. Continuous Monitoring	• Real-time monitoring with CloudWatch • SageMaker Model Monitor for quality checks • Set up alerts for key metrics
2. Automated Remediation	• Auto-scaling policies for endpoints • Automated model retraining triggers • AWS Lambda for automated responses
3. Data Quality Assurance	• Monitor input data drift • Implement data validation checks • Use Amazon Athena for ad-hoc queries
4. Model Performance Tracking	• Track accuracy, latency, throughput • A/B testing for model comparisons • SageMaker Experiments for version logging
5. Explainability and Interpretability	• SageMaker Clarify for bias detection • SHAP values for interpretability • Maintain model cards for documentation
6. Security and Compliance	Encryption at rest and in transit /IAM roles / audit with AWS CloudTrail
7. Cost Optimization	Monitor and optimize resource utilization/ Auto-scaling /Use Spot Instances
8. Scalability and Elasticity	Horizontal scaling/Multi-model endpoints for efficiency/Caching strategies
9. Fault Tolerance and High Availability	Multi AZs/Circuit breakers and fallbacks/Use multi-model endpoints
10. Operational Excellence	IaC with CloudFormation/ AWS Step Functions for ML workflows

How to use AWS CloudTrail to log, monitor, and invoke re-training activities

Aspect	Description	Key Points
Logging	Record API calls and events	...
Monitoring	Track ML-related activities	...
Re-training Triggers	Use events to initiate re-training	• Set up CloudWatch Events rules based on CloudTrail logs • Trigger Lambda functions for automated re-training • Integrate with Step Functions for complex workflows

Monitoring and observability tools to troubleshoot latency and performance issues (for example, AWS X-Ray, Amazon CloudWatch Lambda Insights, Amazon CloudWatch Logs Insights)

Tool	Key Features	Use Cases	Benefits for Troubleshooting
AWS X-Ray	• Distributed tracing • Service map visualization • Trace analysis • Integration with many AWS services	• End-to-end request tracking • Identifying bottlenecks • Analyzing service dependencies	• Visualize application's component interactions • Pinpoint exact location of performance issues • Understand downstream impact of issues
CloudWatch Lambda Insights	ONLY FOR Lambda functions		
CloudWatch Logs Insights	• Log query and visualization • Built-in and custom queries	• ...	• ....

Rightsizing instance - SageMaker Inference Recommender vs AWS Compute Optimizer)

Tool	Purpose	Key Features	Benefits
SageMaker Inference Recommender	Optimize ML model deployment	• Automated benchmarking • Instance type recommendations • Performance vs. cost analysis	• Improved inference performance • Cost optimization for ML workloads
AWS Compute Optimizer	Optimize EC2 instance types	• ML-powered recommendations • Right-sizing suggestions	Performance/Savings boost

Appendix:

Analytics Tools Summary

Service	Description	Primary ML Use Case
Amazon Athena	Serverless query for S3	Ad-hoc analysis of ML datasets
Amazon EMR	Managed big data platform	Large-scale data processing for ML
AWS Glue	Serverless data integration service	ETL for ML data preparation
AWS Glue DataBrew	Visual data preparation tool	Feature engineering and data cleaning
AWS Glue Data Quality	Automated data quality checks	Ensuring ML data quality and consistency
Amazon Kinesis	Real-time data streaming platform	Stream processing for ML applications
Amazon Kinesis Data Firehose	Real-time streaming data delivery	Ingesting streaming data for ML models
AWS Lake Formation	Centralized data lake service	Building secure data lakes for ML
Amazon Managed Service for Apache Flink	Serverless Apache Flink applications	Real-time data processing for ML
Amazon OpenSearch Service	Distributed search and analytics	Log analytics and ML model monitoring
Amazon QuickSight	Business intelligence service	Visualizing ML insights and predictions
Amazon Redshift	Data warehousing service	Large-scale data analysis for ML

AWS Secrets Manager

AWS Secrets Manager is a secrets management service that helps you protect access to your applications, services, and IT resources.

Key Features:

- Secure Storage: Encrypts and stores secrets (e.g., passwords, API keys)
- Rotation: Automates the rotation of secrets
- Fine-grained Access Control: Uses IAM policies to control access
- Auditing: Integrates with AWS CloudTrail for auditing
- Cross-Region Replication: Supports replication of secrets across regions

AWS Storage Gateway

AWS Storage Gateway is a hybrid storage service that enables on-premises applications to seamlessly use AWS cloud storage.

By using AWS Storage Gateway, organizations can seamlessly integrate their on-premises IT environments with AWS cloud storage, enabling hybrid cloud use cases and facilitating cloud migration strategies

Machine Learning:

Service	Primary Function	Key Features/Use Cases
Amazon Augmented AI (A2I)	Human review of ML predictions	• Improves ML model accuracy • Customizable human review workflows • Integrates with SageMaker and other AWS services
Amazon Bedrock	Foundation model service	• Access to pre-trained foundation models • Customization and fine-tuning capabilities • Secure and scalable deployment
Amazon CodeGuru	• Automated code reviews • application performance recommendations	• Identifies code defects and vulnerabilities • Provides performance optimization suggestions • Supports Java and Python
Amazon Comprehend	Natural Language Processing (NLP)	• Entity recognition • Sentiment analysis • Topic modeling • Language detection
Amazon Comprehend Medical	NLP for healthcare and life sciences	• Medical entity extraction • Protected PHI detection
Amazon DevOps Guru	ML-powered cloud operations	• Anomaly detection in operational data • Root cause analysis • Proactive issue resolution recommendations