AWS ML Associate — Full Study Guide (Sections 1.1 to 4.3)

# Section 1: Data Engineering for ML

## 1.1 Collect, Ingest, and Store Data

• Sources: Batch (CSV, Parquet, JSON, logs), Databases (RDS/Aurora, DynamoDB), Streaming (IoT, clicks), External (SFTP, SaaS)

• Ingestion: Kinesis (streams), Glue (ETL), DMS (migrations), Transfer Family (SFTP), Snowball (bulk)

• Storage: S3 (hub for ML), Redshift, RDS, DynamoDB

• Catalog & Query: Glue Data Catalog, Athena, Redshift Spectrum, Lake Formation

• Security & Governance: IAM, SSE-S3/SSE-KMS, VPC endpoints/PrivateLink

• SageMaker: Input channels (Pipe vs File), Data Wrangler, Ground Truth, Model Registry

Exam Tips: Always think S3 → Glue → Athena/Redshift → SageMaker; prefer Pipe for large datasets; encrypt by default

## 1.2 Transform and Prepare Data

• Tools: Glue DataBrew (no-code), Glue ETL (large scale), SageMaker Processing, Data Wrangler

• Concepts: Cleaning (missing values, duplicates), Feature Engineering (encoding, scaling, embeddings), Splits (train/val/test), Data Quality (imbalance handling)

• SageMaker: Processing jobs, Pipelines, Feature Store

Exam Tips: Data Wrangler preferred, stratified sampling for imbalance, use Feature Store for consistency

## 1.3 Train Models

• Training: Built-in algorithms (XGBoost, Linear Learner), Pre-built containers (TF, PyTorch), Custom containers, Autopilot

• Concepts: Input data from S3, choose algorithm per data type, hyperparameters, distributed training (Horovod, parameter servers)

• Tracking: SageMaker Experiments, MLflow integration, metrics (accuracy, precision, recall, RMSE, etc.)

Exam Tips: S3 is the source, built-in algos scale best, Autopilot for baselines, match metric to problem

# Section 2: Modeling

## 2.1 Choose a Modeling Approach

• Classification: Logistic Regression, XGBoost, Linear Learner, DL (CNN/RNN)

• Regression: Linear Regression, XGBoost, DL dense nets

• Clustering: K-Means, PCA for dimensionality reduction

• Recommendation: Factorization Machines, Neural CF

Exam Tips: Factorization Machines = sparse/recs, K-Means = clustering, PCA = dimension reduction

## 2.2 Feature Engineering and Selection

• Engineering: Encoding (one-hot, embeddings), Scaling (min-max, z-score), Text (TF-IDF, embeddings), Images (pixels, transfer learning)

• Selection: Filters (correlation, chi-square), Wrappers (RFE), Embedded (L1/Lasso, tree importance), Dimensionality reduction (PCA)

• Tools: Data Wrangler, Processing jobs, Feature Store

Exam Tips: Scale numeric for distance-based algos, one-hot encoding for categoricals, PCA trades off interpretability

## 2.3 Train and Evaluate Models

• Training: Split data (holdout, CV), run SageMaker training jobs, distributed training

• Metrics: Classification (accuracy, precision, recall, F1, ROC-AUC); Regression (MSE, RMSE, MAE, R²); Clustering (silhouette); Recommendation (precision@k, recall@k)

• Tracking: SageMaker Experiments, log parameters, metrics, artifacts

Exam Tips: Choose metric per task, beware imbalance, prefer CV over single split

## 2.4 Tune Models

• Hyperparameters: LR, depth, reg strength

• Methods: Grid search (exhaustive), Random search (baseline), Bayesian optimization (efficient)

• Tools: SageMaker Automatic Model Tuning, Autopilot

• Practices: define objective metric, avoid overfitting, use early stopping

Exam Tips: Hyperparams ≠ learned weights, Bayesian = most efficient, validate to avoid overfit

# Section 3: Deployment & Operations

## 3.1 Deploy Models

• Options: Real-time endpoints, Batch Transform, Asynchronous inference, Edge (SageMaker Neo)

Exam Tips: Real-time = low latency, Batch = offline large sets, Async = long jobs, Neo = edge devices

## 3.2 Monitor Models

• Tools: SageMaker Model Monitor (drift), CloudWatch, Clarify (bias, explainability)

Exam Tips: Model Monitor = drift, Clarify = fairness/explainability, CloudWatch = metrics/logs

## 3.3 Automate ML Workflows

• Tools: SageMaker Pipelines (native CI/CD), Step Functions (orchestration), CodePipeline/CodeBuild

Exam Tips: Pipelines = ML-native, Step Functions = orchestrate services, CodePipeline = CI/CD integration

# Section 4: Responsible, Secure, and Optimize

## 4.1 Apply Responsible AI Practices

• Practices: Bias detection (Clarify), Explainability (SHAP in Clarify), Governance (lineage, documentation)

Exam Tips: Clarify for bias + explainability, key for regulated industries

## 4.2 Secure ML Solutions

• Practices: Encryption (SSE-S3, SSE-KMS, TLS), IAM least privilege, Networking (VPC, PrivateLink)

Exam Tips: Encrypt by default, least privilege IAM, isolate jobs in VPC

## 4.3 Optimize ML Workloads

• Techniques: Spot instances (cheap, checkpoint!), right-size instances, Debugger for profiling, distributed training

Exam Tips: Spot = save cost but checkpoint, Debugger = optimize, GPU vs CPU = choose wisely