AWS Machine Learning Specialty (MLS-C01) Study Guide

Exam Overview

Attribute	Details
Exam Code	MLS-C01
Duration	180 minutes
Questions	65 questions
Passing Score	750/1000 (approximately 75%)
Format	Multiple choice, multiple response
Cost	$300 USD

Domain Weightings

Domain	Weight
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: ML Implementation and Operations	20%

Domain 1: Data Engineering (20%)

1.1 Data Ingestion Services

Amazon Kinesis Family

Service	Purpose	Key Features
Kinesis Data Streams	Real-time data streaming	Shards for throughput, 24hr-7day retention, real-time processing
Kinesis Data Firehose	Data delivery to destinations	Near real-time, auto-scaling, built-in transformations, delivers to S3/Redshift/Elasticsearch
Kinesis Data Analytics	Real-time SQL analytics	SQL queries on streaming data, integrates with Streams and Firehose
Kinesis Video Streams	Video streaming	Real-time video processing, ML integration

Key Concept: Kinesis Data Streams → Kinesis Data Firehose → S3 is the standard architecture for streaming data ingestion for ML training.

AWS Glue

- **Purpose**: Serverless ETL (Extract, Transform, Load) service
- **Components**:
  - **Glue Data Catalog**: Centralized metadata repository
  - **Glue Crawlers**: Automatically discover and catalog data schemas
  - **Glue Jobs**: ETL jobs using PySpark or Python shell
  - **Glue Studio**: Visual ETL job builder
- **Key Features**:
  - Serverless - no infrastructure management
  - Supports batch and micro-batch processing
  - Job bookmarks track processed data to avoid reprocessing
  - **Sensitive Data Detection transform** for PII detection and masking
  - Dynamic partition overwrite for idempotent reprocessing

AWS Database Migration Service (DMS)

- **Purpose**: Migrate databases to AWS, replicate ongoing changes
- **Key Feature**: Change Data Capture (CDC) for continuous replication
- **ML Use Case**: Replicate OLTP data to S3 for ML training without impacting production databases

1.2 Data Storage

Amazon S3 (Simple Storage Service)

- **Primary data lake storage** for ML workloads
- **File Formats for ML**:

Format	Best For	Characteristics
Parquet	Analytical queries	Columnar, compressed, schema-aware
ORC	Hive workloads	Columnar, optimized for Hadoop
CSV	Simple tabular data	Human-readable, larger file sizes
JSON	Semi-structured data	Flexible schema, larger than Parquet
RecordIO	SageMaker training	Optimized for SageMaker algorithms

- **S3 Storage Classes**: Standard, Intelligent-Tiering, Glacier (for archival)
- **S3 Select**: Query subset of data using SQL (reduces data transfer)

Amazon SageMaker Feature Store

- **Purpose**: Centralized repository for ML features
- **Two Stores**:
  - **Online Store**: Low-latency feature retrieval (single-digit ms) for real-time inference
  - **Offline Store**: S3-based storage for batch training
- **Key Benefits**: Feature versioning, lineage tracking, automatic sync between online/offline

1.3 Data Transformation & Processing

SageMaker Data Wrangler

- Visual interface for data preparation
- 300+ built-in transformations
- Export to SageMaker Processing, Pipelines, or Feature Store

SageMaker Processing

- Run data processing workloads (preprocessing, postprocessing, evaluation)

- Supports Scikit-learn, Spark, and custom containers

Amazon EMR (Elastic MapReduce)

- Managed Hadoop/Spark clusters
- **Use when**: Complex transformations, custom libraries, massive scale (500TB+)
- **Bootstrap actions**: Install custom dependencies
- **Not serverless** - requires cluster management

AWS Batch

- Managed batch computing
- **GPU support** for compute-intensive workloads
- **Use when**: Video processing, GPU-accelerated workloads

1.4 SageMaker Data Input Modes

Mode	Description	Best For
File Mode	Downloads data to local storage before training	Small-medium datasets
Pipe Mode	Streams data directly from S3	Large datasets, faster startup
FastFile Mode	POSIX-compatible file system access	Random access patterns

Key Concept: Pipe mode significantly reduces training startup time for large datasets.

1.5 Data Lake Optimization

- **Partitioning**: Organize data by frequently filtered columns (date, region)
- **Partition Pruning**: Query only relevant partitions
- **Z-Ordering/Data Clustering**: Co-locate related data for faster scans
- **Compaction**: Combine small files into larger ones

Domain 2: Exploratory Data Analysis (24%)

2.1 Data Exploration Tools

Amazon Athena

- **Purpose**: Serverless SQL queries on S3 data
- **Pricing**: Pay per query (based on data scanned)
- **Best For**: Ad-hoc analysis, cost-effective exploration
- **Integration**: Works with Glue Data Catalog for schema management

Amazon QuickSight

- Business intelligence and visualization
- ML-powered insights (anomaly detection, forecasting)

2.2 Handling Missing Data

Method	When to Use
Deletion	Small percentage missing, MCAR, large dataset
Mean Imputation	Numerical, normally distributed, MCAR
Median Imputation	Numerical with outliers or skewed distribution
Mode Imputation	Categorical variables
KNN Imputation	When similar records can inform missing values
Multiple Imputation	When preserving variance is critical

Missing Data Types:

- **MCAR** (Missing Completely at Random): No pattern, safe to impute
- **MAR** (Missing at Random): Depends on observed data
- **MNAR** (Missing Not at Random): Depends on unobserved data, problematic

2.3 Handling Outliers

Method	Description
Winsorization	Cap values at specified percentiles (e.g., 5th/95th)
Trimming	Remove extreme values
Log Transformation	Compress range of large values

Method	Description
Z-score Filtering	Remove values beyond N standard deviations
IQR Method	Remove values beyond 1.5 × IQR from quartiles

2.4 Feature Engineering

Numerical Transformations

Transformation	Purpose	When to Use
Standardization (Z-score)	Mean=0, SD=1	Algorithms sensitive to scale (SVM, KNN, neural nets)
Min-Max Scaling	Scale to [0,1] range	When bounded range needed
Log Transformation	Reduce right skewness	Right-skewed data, large value ranges
Log(x+1)	Handle zeros in log transform	Zero-inflated data with large range
Box-Cox	Normalize distribution	Various skewness patterns
Square Root	Reduce moderate skewness	Count data, mild skewness

Categorical Encoding

Method	Description	When to Use
One-Hot Encoding	Binary column per category	Few categories (<10-15)
Label Encoding	Integer per category	Ordinal data (has natural order)
Target Encoding	Replace with target mean	High cardinality, tree models
Binary Encoding	Binary representation	Medium cardinality
Frequency Encoding	Replace with frequency count	When frequency is informative

Warning: Label encoding for nominal (non-ordinal) data implies false ordering!

Time-Based Features

Feature	Description
Cyclical Encoding	Sin/cos transformation for hour, day, month
Lag Features	Previous time period values
Rolling Statistics	Moving averages, rolling sums
Time Since Event	Duration since last occurrence

Key Concept: Use sine/cosine encoding for cyclical features (hour 23 and hour 1 should be close).

```python
# Cyclical encoding example
hour_sin = np.sin(2 * np.pi * hour / 24)
hour_cos = np.cos(2 * np.pi * hour / 24)
```

2.5 Text Feature Engineering

TF-IDF (Term Frequency-Inverse Document Frequency)

- **TF (Term Frequency)**: How often a term appears in a document
- **IDF (Inverse Document Frequency)**: How rare a term is across corpus
- **Formula**: TF-IDF = TF × log(N / df)
  - N = total documents
  - df = documents containing term
- **Purpose**: Weight distinctive terms higher than common words

Other Text Representations

Method	Description
Bag of Words	Word count vectors (loses word order)
N-grams	Sequences of N consecutive words
Word Embeddings	Dense vector representations (Word2Vec, GloVe)
BERT Embeddings	Contextual embeddings from transformer models

2.6 Dimensionality Reduction

Technique	Type	Preserves	Use Case
PCA	Linear	Global variance	General dimensionality reduction
t-SNE	Non-linear	Local neighborhoods	2D/3D visualization
UMAP	Non-linear	Local + global structure	Visualization, clustering
LDA	Supervised	Class separation	Classification preprocessing

PCA Key Points:

- Creates new features (principal components) as linear combinations
- Components are orthogonal (uncorrelated)
- First component captures maximum variance
- Not interpretable (transformed features)

2.7 Feature Selection

Method	Type	Description
Variance Threshold	Filter	Remove low-variance features
Correlation Analysis	Filter	Remove highly correlated features
Mutual Information	Filter	Statistical dependency with target (no model needed)
Chi-Square Test	Filter	For categorical features
RFE	Wrapper	Recursively remove least important features
L1 Regularization	Embedded	Zero out unimportant coefficients
Feature Importance	Embedded	Tree-based model importance scores

2.8 Class Imbalance

Technique	Description
Oversampling	Duplicate minority class samples

Technique	Description
SMOTE	Generate synthetic minority samples by interpolation
Undersampling	Remove majority class samples
Class Weights	Penalize misclassification of minority class more
Threshold Adjustment	Lower classification threshold for minority class

Warning: Increasing learning rate does NOT address class imbalance!

2.9 Correlation and Multicollinearity

- **Pearson Correlation**: Linear relationship (-1 to 1)
- **Multicollinearity**: High correlation between features
- **Problem in Linear Models**: Unstable coefficients, inflated standard errors
- **Detection**: VIF (Variance Inflation Factor) > 5 or 10
- **Solutions**: Remove one of correlated features, PCA, regularization

---

Domain 3: Modeling (36%)

3.1 SageMaker Built-in Algorithms

Supervised Learning - Classification/Regression

Algorithm	Type	Use Case	Key Parameters
XGBoost	Ensemble	Tabular data, competitions	max_depth, eta, num_round, subsample
Linear Learner	Linear	Binary/multiclass, regression	predictor_type, learning_rate
K-Nearest Neighbors	Instance-based	Classification, regression	k, sample_size
Factorization Machines	Matrix factorization	Recommendations, sparse data	num_factors

Unsupervised Learning

Algorithm	Type	Use Case
K-Means	Clustering	Group similar data points
PCA	Dimensionality reduction	Reduce features, preserve variance
Random Cut Forest (RCF)	Anomaly detection	Detect outliers in streaming data
IP Insights	Anomaly detection	Detect suspicious IP behavior

Deep Learning

Algorithm	Use Case	Framework
Image Classification	Classify images	MXNet
Object Detection	Detect objects with bounding boxes	MXNet
Semantic Segmentation	Pixel-level classification	MXNet
BlazingText	Text classification, Word2Vec	Custom
Sequence-to-Sequence	Translation, summarization	MXNet
Neural Topic Model	Topic modeling	MXNet
DeepAR	Time series forecasting	MXNet
Object2Vec	Embeddings for pairs	MXNet

3.2 Algorithm Selection Guide

Problem Type	Algorithms
Binary Classification	XGBoost, Linear Learner, KNN
Multi-class Classification	XGBoost, Linear Learner, KNN
Regression	XGBoost, Linear Learner, KNN
Count Prediction	Poisson Regression
Anomaly Detection	Random Cut Forest, IP Insights

Problem Type	Algorithms
Clustering	K-Means
Recommendations	Factorization Machines
Time Series Forecasting	DeepAR, Prophet, ARIMA
Text Classification	BlazingText, BERT fine-tuning
Object Detection	SSD, YOLO, Faster R-CNN
Image Classification	ResNet, VGG, custom CNN

3.3 Model Training Concepts

Overfitting vs Underfitting

Condition	Training Loss	Validation Loss	Solution
Overfitting	Low	High (or increasing)	More data, regularization, early stopping, dropout
Underfitting	High	High	More complexity, more features, longer training
Good Fit	Low	Low (similar to training)	Deploy!

Regularization Techniques

Technique	Description	Effect
L1 (Lasso)	Adds absolute value of coefficients to loss	Sparse solutions, feature selection
L2 (Ridge)	Adds squared coefficients to loss	Smaller coefficients, prevents overfitting
Elastic Net	Combination of L1 and L2	Balance of both
Dropout	Randomly zero out neurons during training	Prevents co-adaptation
Early Stopping	Stop when validation loss increases	Prevents overtraining
Batch Normalization	Normalize layer inputs	Stabilizes training, allows higher learning rates

XGBoost Hyperparameters for Overfitting

To Reduce Overfitting	Adjustment
`max_depth`	Decrease (shallower trees)
`min_child_weight`	Increase (more samples per leaf)
`subsample`	Decrease (use subset of data per tree)
`colsample_bytree`	Decrease (use subset of features)
`lambda` (L2)	Increase
`alpha` (L1)	Increase
`num_round`	Decrease (fewer trees)

3.4 Hyperparameter Tuning

SageMaker Automatic Model Tuning

Strategy	Description	Best For
Grid Search	Try all combinations	Small search spaces
Random Search	Random sampling	Larger spaces, quick exploration
Bayesian Optimization	Learn from previous trials	Expensive evaluations, large spaces

Bayesian Optimization: Builds probabilistic model, intelligently selects next hyperparameters based on previous results.

3.5 Evaluation Metrics

Classification Metrics

Metric	Formula	When to Use
Accuracy	(TP+TN)/(TP+TN+FP+FN)	Balanced classes
Precision	TP/(TP+FP)	Minimize false positives
Recall (Sensitivity)	TP/(TP+FN)	Minimize false negatives
F1 Score	2×(P×R)/(P+R)	Balance precision/recall
AUC-ROC	Area under ROC curve	Ranking, probability calibration
PR-AUC	Area under Precision-Recall curve	Imbalanced datasets

Key Insight: High ROC-AUC + Low PR-AUC = Severely imbalanced dataset

Business Cost Examples:

- **Fraud Detection** (missing fraud costly): Prioritize **Recall**
- **Email Spam** (false positives annoying): Balance **Precision/Recall**
- **Medical Diagnosis** (missing disease dangerous): Prioritize **Recall**
- **Legal Discovery** (false positives expensive): Prioritize **Precision**

Regression Metrics

Metric	Description
MSE	Mean Squared Error - penalizes large errors
RMSE	Root MSE - same units as target
MAE	Mean Absolute Error - robust to outliers
R^2	Proportion of variance explained
MAPE	Mean Absolute Percentage Error

Object Detection Metrics

Metric	Description
IoU	Intersection over Union - bounding box overlap
mAP	Mean Average Precision - standard metric
Precision@IoU	Precision at specific IoU threshold

3.6 Neural Network Concepts

Activation Functions

Function	Range	Use Case
ReLU	$[0, \infty)$	Hidden layers (default choice)
Sigmoid	(0, 1)	Binary output, multi-label
Softmax	(0, 1), sum=1	Multi-class output (mutually exclusive)
Tanh	(-1, 1)	Hidden layers, centered output

Multi-label vs Multi-class:

- **Multi-class**: One label per sample → **Softmax**
- **Multi-label**: Multiple labels per sample → **Sigmoid** with binary cross-entropy

Optimizers

Optimizer	Description
SGD	Basic gradient descent
SGD + Momentum	Accelerates SGD in consistent direction
Adam	Adaptive learning rates (most popular)
RMSprop	Adaptive learning rates

Batch Size Effects

Batch Size	Training	Generalization
Small (1-32)	Noisy gradients, slow	Often better
Large (256+)	Stable gradients, fast	May be worse

3.7 Transfer Learning

- **Concept**: Use pre-trained model as starting point
- **When to Use**: Limited labeled data, similar domain
- **Process**:
    1. Take pre-trained model (e.g., BERT, ResNet)
    2. Replace final layer(s) for new task
    3. Fine-tune on new data (often with lower learning rate)

Key Insight: Pre-trained language models (BERT) are highly effective for NLP tasks with limited data.

3.8 Recommendation Systems

Type	Data Needed	How It Works
Content-Based	Item features/metadata	Recommend similar items
Collaborative Filtering	User-item interactions	Find similar users/items by behavior
Hybrid	Both	Combine approaches

Cold Start Problem: New users/items have no history → Use content-based or hybrid

3.9 Time Series Forecasting

Model	Seasonality	Trend	Use Case
Simple Exponential Smoothing	No	No	Level only
Holt's	No	Yes	Trend
Holt-Winters	Yes	Yes	Trend + seasonality
ARIMA	No	Yes	General time series
SARIMA	Yes	Yes	Seasonal time series

Model	Seasonality	Trend	Use Case
Prophet	Yes (multiple)	Yes	Business forecasting, robust
DeepAR	Yes	Yes	Multiple related series

3.10 Common Pitfalls

Data Leakage

- **Definition**: Training data contains information not available at prediction time
- **Examples**:
  - Future information in features
  - Target variable encoded in features
  - Incorrect train/test split (random split on time series)
- **Prevention**: Use temporal splits for time data, careful feature engineering

Temporal Data Leakage

- **Problem**: Random train/test split on time series data
- **Result**: Model sees future data during training
- **Solution**: Always use temporal split (train on past, test on future)

Domain 4: ML Implementation and Operations (20%)

4.1 SageMaker Deployment Options

Option	Latency	Scaling	Cost Model	Use Case
Real-time Inference	Low (ms)	Auto-scaling	Per instance-hour	Production APIs
Serverless Inference	Medium	Auto (to zero)	Per request	Variable/sparse traffic
Batch Transform	High	Job-based	Per job	Large batch processing
Asynchronous Inference	Medium	Queue-based	Per request	Large payloads, long processing

Key Insight: Serverless Inference scales to zero = no cost during idle time.

4.2 Deployment Strategies

Strategy	Description	Rollback
Blue/Green	Run two versions, shift traffic	Instant
Canary	Gradual traffic shift (10% → 50% → 100%)	Instant
Rolling	Replace instances gradually	Slower
A/B Testing	Route specific users to variants	Instant

SageMaker Production Variants: Native support for blue/green and canary deployments.

4.3 Model Monitoring

Amazon SageMaker Model Monitor

Monitor Type	What It Detects
Data Quality	Schema changes, missing values, data type changes
Model Quality	Accuracy degradation, metric drift
Bias Drift	Changes in fairness metrics
Feature Attribution Drift	Changes in feature importance

Concept Drift / Data Drift

- **Data Drift**: Input data distribution changes over time
- **Concept Drift**: Relationship between inputs and outputs changes
- **Detection**: Compare current data statistics to baseline
- **Solution**: Retrain on recent data

4.4 Model Optimization

Amazon SageMaker Neo

- **Purpose**: Compile models for target hardware
- **Benefits**:
  - 2-10x performance improvement
  - Reduced model size

    - Hardware-specific optimizations
- **Supported**: TensorFlow, PyTorch, MXNet, XGBoost, ONNX

Inference Optimization Techniques

Technique	Description
Model Compilation (Neo)	Optimize for target hardware
Quantization	Reduce precision (FP32 → INT8)
Pruning	Remove unnecessary weights
Knowledge Distillation	Train smaller model from larger one
GPU Inference	Use GPU instances for deep learning

4.5 MLOps and CI/CD

SageMaker Pipelines

- **Purpose**: Orchestrate ML workflows
- **Features**:
    - Define steps: Processing, Training, Evaluation, Registration
    - Automatic lineage tracking
    - Integration with Model Registry

SageMaker Model Registry

- **Purpose**: Version and manage models
- **Features**:
    - Model versioning
    - Approval workflows (Pending → Approved → Rejected)
    - Deployment tracking
    - Metadata and lineage

Typical MLOps Pipeline

```
Code Change → Build → Train → Evaluate → Register → Approve → Deploy
                ↓       ↓         ↓
           SageMaker  SageMaker   Model
           Processing  Training   Registry
```

4.6 Model Explainability

Amazon SageMaker Clarify

- **Pre-training Bias Detection**: Analyze training data for bias
- **Post-training Bias Detection**: Analyze model predictions for bias
- **Feature Attributions**: SHAP values for individual predictions

SHAP (SHapley Additive exPlanations)

- **Purpose**: Explain individual predictions
- **Output**: Contribution of each feature to prediction
- **Use Case**: Regulatory compliance (lending, healthcare)

4.7 Security and Compliance

Data Protection

Feature	Purpose
VPC Endpoints	Private connectivity to SageMaker
Encryption at Rest	KMS encryption for S3, EBS
Encryption in Transit	TLS for all communications
IAM Roles	Control access to resources
Network Isolation	No internet access for training

Amazon Macie

- Discover and protect sensitive data in S3
- Automated PII detection
- Security monitoring (not for ML bias)

4.8 Cost Optimization

Strategy	How
Spot Instances	Up to 90% savings for training (not real-time inference)
Serverless Inference	Pay only for actual requests
Right-sizing	Choose appropriate instance types
Auto-scaling	Scale down during low traffic
Multi-model Endpoints	Multiple models on single endpoint

4.9 Distributed Training

Data Parallelism

- **Concept**: Split data across multiple GPUs/instances
- **When**: Large datasets, model fits in single GPU
- **Frameworks**: Horovod, PyTorch DDP, SageMaker Data Parallel

Model Parallelism

- **Concept**: Split model across multiple GPUs
- **When**: Model too large for single GPU
- **Framework**: SageMaker Model Parallel

Key Insight: Multi-GPU training requires explicit configuration in training script!

4.10 Inference Pipeline

- **Purpose**: Chain preprocessing, inference, and postprocessing
- **Use Cases**:
  - Input validation and transformation
  - Feature engineering at inference time
  - Output validation (enforce business rules)
  - Multiple model ensemble

AWS Services Quick Reference

ML-Specific Services

Service	Purpose
SageMaker	Full ML platform (build, train, deploy)
SageMaker Ground Truth	Data labeling
SageMaker Feature Store	Feature management
SageMaker Clarify	Bias detection, explainability
SageMaker Model Monitor	Production monitoring
SageMaker Neo	Model compilation/optimization
SageMaker Debugger	Training debugging
SageMaker Experiments	Experiment tracking
Comprehend	NLP (sentiment, entities, PII)
Rekognition	Image/video analysis
Textract	Document text extraction
Transcribe	Speech to text
Polly	Text to speech
Translate	Language translation
Forecast	Time series forecasting
Personalize	Recommendations
Fraud Detector	Fraud detection
Lex	Conversational AI
Kendra	Intelligent search

Data Services

Service	Purpose
S3	Object storage (data lake)
Glue	Serverless ETL, Data Catalog
Athena	Serverless SQL queries on S3
EMR	Managed Hadoop/Spark
Kinesis	Real-time streaming
Redshift	Data warehouse
DynamoDB	NoSQL database
ElastiCache	In-memory caching (Redis/Memcached)
DMS	Database migration/replication

Exam Tips

Key Patterns to Recognize

1. **"Streaming data for ML training"** → Kinesis Data Streams → Firehose → S3
2. **"Large dataset, slow training startup"** → SageMaker Pipe mode
3. **"Feature store for real-time and batch"** → SageMaker Feature Store
4. **"Serverless ETL"** → AWS Glue
5. **"Ad-hoc queries on S3"** → Amazon Athena
6. **"Anomaly detection in streaming data"** → Random Cut Forest
7. **"Explain individual predictions"** → SageMaker Clarify (SHAP)
8. **"Bias detection"** → SageMaker Clarify
9. **"Model performance degraded over time"** → Data/concept drift → Retrain
10. **"Variable traffic, cost optimization"** → Serverless Inference
11. **"Zero-downtime deployment"** → Blue/green with production variants
12. **"Low-latency inference optimization"** → SageMaker Neo
13. **"Class imbalance"** → SMOTE, class weights, threshold adjustment (NOT learning rate)

14. **"High cardinality categorical"** → Target encoding
15. **"Right-skewed data"** → Log transformation

Common Traps

1. **Accuracy on imbalanced data**: High accuracy can be misleading
2. **Random split on time series**: Causes temporal leakage
3. **Label encoding for nominal data**: Implies false ordinal relationship
4. **Log(0)**: Undefined - use log(x+1)
5. **Softmax for multi-label**: Wrong - use sigmoid
6. **Spot instances for inference**: Not supported for real-time endpoints
7. **Increasing learning rate for instability**: Makes it worse
8. **More features for class imbalance**: Doesn't help
9. **Mean imputation for skewed data**: Use median instead
10. **PCA for interpretability**: Components are not interpretable

Study Priorities by Weight

1. **Modeling (36%)**: Know algorithms, metrics, overfitting/underfitting, hyperparameters
2. **EDA (24%)**: Know transformations, encoding, feature engineering, class imbalance
3. **Data Engineering (20%)**: Know Kinesis, Glue, S3, Feature Store, input modes
4. **MLOps (20%)**: Know deployment options, monitoring, Clarify, Neo, Pipelines

---

Practice Checklist

- [ ] Can explain the difference between all Kinesis services
- [ ] Know when to use Pipe mode vs File mode
- [ ] Understand TF-IDF components
- [ ] Can identify overfitting vs underfitting from loss curves
- [ ] Know which metrics to use for imbalanced classification
- [ ] Understand SMOTE and class weights
- [ ] Can choose between encoding methods for categorical data
- [ ] Know SageMaker built-in algorithms and their use cases
- [ ] Understand XGBoost hyperparameters for reducing overfitting
- [ ] Can explain transfer learning benefits
- [ ] Know deployment options and when to use each

- [ ] Understand data drift and concept drift
- [ ] Can explain SHAP values and SageMaker Clarify
- [ ] Know Model Monitor capabilities
- [ ] Understand blue/green deployments

Good luck on your exam!