AWS Certified Machine Learning Engineer - Associate Exam Guide

# Section 2.1 — Choose a Modeling Approach (In-Depth)

This section focuses on identifying the correct modeling approach for a given ML problem. It covers problem framing, algorithm selection, interpretability considerations, and AWS services to accelerate the process.

## 1. Problem Framing

• Classification → predict discrete labels (binary, multi-class). Examples: fraud detection, sentiment analysis.

• Regression → predict continuous values. Examples: price prediction, demand forecasting.

• Clustering → group similar items without labels. Examples: customer segmentation, anomaly detection.

• Recommendation systems → suggest items. Examples: product recommendations, personalization.

## 2. Algorithm Selection

• XGBoost → tree-based, works well on tabular data

• Linear Learner → regression/classification, fast and interpretable

• K-Means → unsupervised clustering

• Factorization Machines → recommendation problems

• Neural Networks (PyTorch/TensorFlow) → deep learning for text, vision, speech

• Forecasting models → DeepAR, Prophet

## 3. Interpretability Considerations

• Choose simpler models (linear/logistic regression) when explainability is required

• Use tree-based models for feature importance visibility

• Use SageMaker Clarify for post-hoc interpretability (SHAP values)

## 4. AWS Services for Modeling

• SageMaker JumpStart → prebuilt solutions, foundation models, and templates for common use cases

• SageMaker Autopilot → automatically explores algorithms and produces baseline models

• Amazon Bedrock → access to foundation models via API (text, chat, embeddings)

• AI Services (Rekognition, Comprehend, Translate, Polly, Transcribe) → ready-to-use services for vision, NLP, and speech

• SageMaker Built-in Algorithms → optimized implementations for common ML problems

## 5. Cost & Performance Trade-offs

• Simpler models (linear/logistic regression) are cheaper and faster

• Deep learning requires GPU/accelerators and more cost

• Consider cost, latency, and accuracy requirements when selecting

## ✅ Exam Tips

• Frame the problem first: classification, regression, clustering, or recommendation

• JumpStart is often the fastest way to deploy a baseline solution

• Autopilot = automatic baseline model generation and tuning

• Bedrock is for generative AI use cases (text, chat, embeddings)

• Use Clarify when interpretability or fairness is required

• Built-in algorithms are optimized for performance on AWS