# AI/ML Engineer Interview Question Bank

## 1. Random Forest Classifier

Conceptual:

- What is a Random Forest and how does it work?
- How does it differ from a Decision Tree?
- Why does Random Forest reduce overfitting?
- How is feature importance calculated?

Practical:

- When would you choose Random Forest over Gradient Boosting?
- How do you tune parameters like n_estimators, max_depth, min_samples_split?
- How do you handle class imbalance with Random Forest?

Code-Based:

- Train and evaluate a Random Forest on a classification problem.
- Visualize and interpret feature importances.

Answers:

- A Random Forest is an ensemble of decision trees where each tree is trained on a random subset of data and features.
- It reduces overfitting by averaging predictions from multiple trees, reducing variance.
- Feature importance is calculated by the decrease in impurity (e.g., Gini or entropy) each feature brings.
- It handles imbalanced data using class weights or balanced sampling.

## 2. XGBoost

Conceptual:

- What is boosting? How does XGBoost implement it?
- Difference between bagging and boosting?
- How does regularization work in XGBoost?
- What makes XGBoost fast and scalable?

Practical:

- Common pitfalls when tuning XGBoost.

- Early stopping - how and why?
- How does XGBoost handle missing values?

Code-Based:

- Train an XGBoost model with GridSearch or RandomSearch.
- Use SHAP for interpretability.

Answers:

- XGBoost is an efficient gradient boosting algorithm using second-order gradients and regularization.
- It uses additive trees, where each new tree fixes residual errors of the previous ones.
- Regularization parameters (lambda, alpha) help prevent overfitting.
- It automatically handles missing values during training.

## 3. LSTM & CNN (Deep Learning)

LSTM:

Conceptual:

- How does LSTM solve the vanishing gradient problem?
- What are input, forget, and output gates?

Practical:

- Use cases for LSTM: time-series, NLP?
- Compare LSTM vs GRU vs Transformer.

Code-Based:

- Build an LSTM in TensorFlow or PyTorch.

CNN:

Conceptual:

- What's a convolution operation?
- Role of pooling, stride, and padding?
- What are feature maps?

Practical:

- CNNs for text - how does that work?

- Transfer learning using CNNs (e.g., ResNet).

Code-Based:

- Train a CNN on image classification using PyTorch or TensorFlow.

Answers:

LSTM:

- LSTM (Long Short-Term Memory) uses gates to control memory updates, solving the vanishing gradient problem.
- GRUs are a simplified version with fewer gates.

CNN:

- Convolutional layers extract spatial features using filters.
- Pooling layers reduce spatial size and computational load.
- CNNs are used in image, audio, and text applications (1D for sequences, 2D for images).

## 4. Hypothesis Testing

Conceptual:

- What is a p-value?
- Null vs alternative hypothesis?
- One-tailed vs two-tailed tests?
- Explain confidence intervals.

Practical:

- Which test to use: A/B test, t-test, chi-square?
- How do you interpret a test result with p=0.06?

Code-Based:

- Implement t-tests, ANOVA, and chi-square in scipy.stats.
- Perform and interpret an A/B test.

Answers:

- Hypothesis testing evaluates evidence against a null hypothesis using p-values.
- Type I error: false positive; Type II error: false negative.
- A p-value < 0.05 typically leads to rejecting the null hypothesis.
- t-test compares means; chi-square tests for independence in categorical data.

## 5. Exploratory Data Analysis (EDA)

Conceptual:

- What are the key steps in EDA?
- Common univariate and multivariate techniques?
- How do you detect skewness and kurtosis?

Practical:

- Strategy for analyzing a large dataset with missing values.
- EDA for time-series vs text vs images.

Code-Based:

- EDA with pandas, seaborn, plotly.
- Correlation heatmaps, pair plots, boxplots.

Answers:

- EDA involves summarizing datasets using plots, stats, and checks.
- Univariate analysis: histograms, boxplots; Bivariate: scatterplots, heatmaps.
- Detect skewness via histograms or skew() function.
- Handle missing values using imputation or dropping.

## 6. Feature Engineering

Conceptual:

- Types of features: categorical, numerical, temporal, text-based.
- What is one-hot encoding vs label encoding?
- Explain feature scaling: normalization vs standardization.

Practical:

- How to handle high-cardinality categorical features?
- Feature engineering for NLP or time-series.

Code-Based:

- Use scikit-learn, category_encoders for feature generation.
- Create lag features for time series.

Answers:

- Involves creating new features from raw data to improve model performance.
- Categorical features: one-hot, label, or target encoding.
- Scaling: normalization (0-1), standardization (mean=0, std=1).
- Time-series: add lag, rolling mean, time-based features.

## 7. RAG-Based Agentic GenAI

RAG Concepts:

Conceptual:

- What is RAG (Retrieval-Augmented Generation)?
- How does it improve over traditional LLM prompting?
- Difference between dense vs sparse retrieval in RAG?
- What are vector stores (e.g., FAISS) and why are they important?

Practical:

- How would you implement a scalable RAG pipeline?
- What are the failure modes of RAG systems?
- How do you evaluate RAG performance?

Code-Based:

- Build a simple RAG pipeline using LangChain or Haystack.
- Index documents using FAISS, then query with LLMs.

Agentic Systems:

Conceptual:

# AI/ML Engineer Interview Question Bank

- What is an AI agent?

- What are tools, memory, and planner-executor patterns?

Practical:

- When would you use agents over standard LLM calls?

- How do you avoid infinite loops in agents?

Code-Based:

- Build an agent using LangChain or OpenAI Assistants API.

Answers:

RAG:

- Combines retrieval (e.g., from FAISS) with LLM generation for grounded answers.

- Dense retrieval uses embeddings; sparse uses keywords (e.g., BM25).

- Evaluation: grounding rate, precision@k, faithfulness.

Agents:

- Agents use tools, memory, and reasoning to perform tasks step-by-step.

- Planner-executor pattern decouples goal decomposition and tool usage.

- LangChain, OpenAI Assistants API support agent workflows.