AI / ML Interview Question & Answer Guide

Mid-Level (2–5 Years) | AI / ML Concepts | General / Mixed Role | 19 Questions

Contents

This guide covers 19 high-frequency AI/ML interview questions with detailed answers and interviewer tips. Topics span core ML theory, deep learning, MLOps, and statistics — the key pillars assessed at the mid-level.

Section 1 — Core ML Concepts

Foundational knowledge every mid-level ML professional must know

Q1. What is the bias-variance trade-off and why does it matter?

Bias is the error from wrong assumptions in the learning algorithm; variance is the error from sensitivity to small fluctuations in training data.

- **High bias** → underfitting (model too simple, misses patterns).
- **High variance** → overfitting (model memorises training data, fails on new data).

The trade-off: reducing bias often increases variance and vice versa. The goal is to find the sweet spot that minimises total error on unseen data.

Pro Tip: *Mention regularisation (L1/L2), cross-validation, or ensemble methods as practical ways to manage this trade-off.*

Q2. Explain the difference between supervised, unsupervised, and reinforcement learning.

- **Supervised Learning:** Model trained on labelled data (input-output pairs). Goal: learn a mapping. Examples: regression, classification.
- **Unsupervised Learning:** No labels provided. Model discovers hidden structure. Examples: clustering (K-Means), dimensionality reduction (PCA).
- **Reinforcement Learning:** Agent learns by interacting with an environment and receiving rewards/penalties. Examples: game-playing AI, robotics.

Pro Tip: *Relate each to a real-world use case you have worked on to make the answer memorable.*

Q3. What is cross-validation and which type do you prefer for tabular data?

Cross-validation (CV) is a technique to evaluate model generalisation by training on subsets and testing on held-out data.

- **K-Fold CV:** Data split into k folds; model trained k times, each time using a different fold as validation. Robust and preferred for medium-to-large tabular datasets.
- **Stratified K-Fold:** Preserves class distribution in each fold — best for imbalanced classification tasks.
- **Leave-One-Out (LOO):** Each sample used once as a test set; computationally expensive but useful for very small datasets.

Pro Tip: *For time-series data, always use TimeSeriesSplit (walk-forward validation) to prevent data leakage.*

Q4. How do you handle class imbalance in a classification problem?

• **Resampling:** Oversample the minority class (SMOTE) or undersample the majority class (random undersampling).
• **Class weights:** Set class_weight='balanced' in sklearn; penalises misclassifying the minority class more.
• **Threshold tuning:** Adjust decision threshold (default 0.5) based on Precision-Recall curve rather than ROC-AUC.
• **Ensemble methods:** BalancedRandomForest, EasyEnsemble combine resampling with boosting.
• **Evaluation:** Use F1-score, PR-AUC, or Matthews Correlation Coefficient instead of raw accuracy.

Pro Tip: *Always mention the evaluation metric alongside the technique — imbalance strategy is meaningless without the right metric.*

Section 2 — Deep Learning & Neural Networks

Architecture knowledge and training best practices

Q5. What is the vanishing gradient problem and how is it mitigated?

During backpropagation, gradients are multiplied layer by layer. With many layers and activation functions like sigmoid/tanh, gradients become exponentially small — weights in early layers barely update.

Common mitigations:

• **ReLU / Leaky ReLU activations:** Do not saturate for positive values, allowing gradients to flow.
• **Batch Normalisation:** Normalises layer inputs, stabilises and accelerates training.
• **Residual connections (ResNets):** Skip connections allow gradients to bypass layers directly.
• **Weight initialisation:** He initialisation for ReLU, Xavier for tanh — prevents gradients exploding or vanishing at start.

Pro Tip: *Mention that exploding gradients (opposite problem) are handled with gradient clipping.*

Q6. Explain the Transformer architecture and why it replaced RNNs for NLP tasks.

The Transformer (Vaswani et al., 2017) uses a self-attention mechanism instead of recurrence.

• **Self-Attention:** Each token attends to every other token simultaneously — captures long-range dependencies that RNNs struggle with.
• **Multi-Head Attention:** Multiple attention heads capture different types of relationships in parallel.
• **Positional Encoding:** Injects sequence order information since there is no inherent ordering.

Advantages over RNNs: fully parallelisable (much faster training on GPUs), better at capturing long-range context, and scales effectively with data and compute.

Pro Tip: *If asked about BERT vs GPT, clarify: BERT uses encoder-only (bidirectional) for understanding tasks; GPT uses decoder-only (autoregressive) for generation.*

Q7. What is dropout and how does it prevent overfitting?

Dropout randomly deactivates a fraction (rate p) of neurons during each training forward pass.

- Prevents neurons from co-adapting — each neuron must learn independently useful features.
- Acts as training an ensemble of 2^n sub-networks, averaging their predictions at inference.
- At inference time, dropout is disabled and weights are scaled by (1-p) to maintain expected output magnitude.

Typical rates: 0.2–0.5 for dense layers; lower or zero for convolutional layers.

Pro Tip: *Mention that with modern networks (e.g., ResNets, Transformers) Batch Norm often reduces the need for high dropout rates.*

Q8. Describe the difference between CNN, RNN, and Transformer — when would you use each?

- **CNN (Convolutional Neural Network):** Excels at spatial data — images, audio spectrograms. Uses local filters with weight sharing. Use for image classification, object detection, semantic segmentation.
- **RNN / LSTM / GRU:** Designed for sequential data with temporal dependencies. Use for time-series forecasting, speech recognition (legacy), simple language modelling.
- **Transformer:** Best for NLP and multimodal tasks. Parallelisable and handles long contexts. Use for text classification, generation, translation, vision (ViT), and multimodal models.

Pro Tip: *For tabular data, tree-based models (XGBoost, LightGBM) often still outperform all three.*

Section 3 — MLOps & Production ML

Deploying, monitoring, and maintaining models at scale

Q9. What is data leakage and how do you prevent it?

Data leakage occurs when information from outside the training dataset — typically from the future or from the test set — influences model training, leading to unrealistically high validation scores that collapse in production.

Common causes:

- **Target leakage:** Features derived from the target or computed after the event being predicted.
- **Train-test contamination:** Scaling/encoding fit on the entire dataset before splitting.

Prevention strategies:

- Always split data before any preprocessing; fit transformers only on training data, then transform validation/test.
- Use Pipelines (sklearn) to encapsulate preprocessing and ensure it is applied correctly at each fold.
- Carefully audit feature engineering logic for temporal ordering.

Pro Tip: *A useful heuristic: if a feature would not be available at prediction time in production, it is a leak.*

Q10. What is model drift and how do you monitor for it in production?

- **Data drift (covariate shift):** Input feature distribution changes over time (e.g., user behaviour shifts). Detect using statistical tests (KS test, PSI) on feature distributions.
- **Concept drift:** The relationship between features and target changes (e.g., fraud patterns evolve). Harder to detect — monitor model performance metrics over time.

Monitoring strategies:

- Log predictions and features in production; compute distribution statistics at regular intervals.
- Set up alerting on performance metrics (accuracy, F1, AUC) with sliding windows.
- Use dedicated tools: Evidently AI, WhyLogs, Arize, or custom dashboards (Grafana).
- Schedule periodic retraining or trigger retraining when drift exceeds a threshold.

Pro Tip: *Mention shadow deployment / A/B testing as a safe way to roll out retrained models before full production switch-over.*

Q11. Explain the difference between batch inference and real-time inference. How do you choose?

- **Batch inference:** Run predictions on large datasets on a schedule (e.g., nightly). Lower cost, higher throughput. Suitable when predictions can be pre-computed (recommendation pre-generation, churn scores).
- **Real-time (online) inference:** Predictions served on-demand with low latency (ms to seconds). Required when the input is not known ahead of time (fraud detection, search ranking, chatbots).

Decision factors:

- Latency requirement: if <100ms needed → real-time (model serving: FastAPI, TorchServe, TF Serving, Triton).
- Volume and cost: batch is cheaper; real-time requires always-on infrastructure.
- Freshness: if stale predictions are acceptable → batch; if not → real-time.

Pro Tip: *A hybrid approach — batch pre-compute for popular items, real-time fallback for cold-start — is common in recommendation systems.*

Q12. How would you approach feature engineering for a structured/tabular dataset?

- **Exploratory analysis first:** Distribution plots, correlation matrix, target analysis to understand the data.
- **Handle missing values:** Mean/median imputation for numerics; mode or 'Unknown' category for categoricals; or model-based imputation.
- **Encode categoricals:** Ordinal encoding for ordered categories; one-hot for low cardinality; target encoding for high cardinality.
- **Create interaction features:** Ratios, products, differences of related features (e.g., debt-to-income ratio).
- **Time-based features:** Extract hour, day-of-week, month, lag features, rolling statistics.
- **Feature selection:** Remove low-variance and high-correlation features; use permutation importance or SHAP values for importance ranking.

Pro Tip: *Automate repetitive FE with tools like Feature-engine or featuretools, but always validate with domain knowledge.*

Section 4 — Statistics & Model Evaluation

Metrics, experimentation, and statistical rigour

Q13. When would you use AUC-ROC vs Precision-Recall AUC? Explain your reasoning.

- **AUC-ROC** measures the ability to discriminate between classes across all thresholds. It is optimistic in the presence of class imbalance because it weighs True Negatives — which are plentiful in imbalanced datasets.
- **PR-AUC (Average Precision)** focuses only on the positive class — directly measures how precisely the model identifies positives at various recall levels.

Use PR-AUC when:

- The positive class is rare (fraud, disease detection).
- The cost of false positives is high.
- Business stakeholders care about Precision and Recall, not the negative class.

Pro Tip: *A model can have AUC-ROC of 0.95 but PR-AUC of 0.30 on a heavily imbalanced dataset — always report both.*

Q14. What is the p-value and what are its common misinterpretations?

The p-value is the probability of observing results at least as extreme as the actual results, assuming the null hypothesis is true.

Common misinterpretations:

- **Wrong:** p-value is the probability that the null hypothesis is true.
- **Wrong:** $p < 0.05$ means the result is practically significant — a tiny effect can be statistically significant with a large sample.
- **Wrong:** 1 - p is the probability the alternative hypothesis is true.

Better practice: report effect size (Cohen's d), confidence intervals, and power alongside p-values.

Pro Tip: *In A/B testing interviews, mention that you also check for novelty effects, multiple testing corrections (Bonferroni, FDR), and minimum detectable effect.*

Q15. Explain regularisation — L1 (Lasso) vs L2 (Ridge). When do you use each?

Regularisation adds a penalty term to the loss function to discourage large weights, reducing overfitting.

- **L1 (Lasso) — $|w|$:** Produces sparse models by driving some weights exactly to zero. Effectively performs feature selection. Use when you suspect many features are irrelevant.
- **L2 (Ridge) — w^2:** Shrinks all weights towards zero but rarely to exactly zero. Handles correlated features better. Use when most features contribute something.
- **Elastic Net:** Combines L1 + L2 — useful when there are groups of correlated features and you still want some sparsity.

Pro Tip: *In tree-based models (XGBoost, LightGBM) regularisation is controlled via max_depth, min_child_weight, and lambda/alpha — conceptually the same goal.*

Bonus — Interview Strategy Tips

How to stand out in your AI/ML interview

✦ Structure your answers with STAR
For behavioural/scenario questions: Situation, Task, Action, Result. Always quantify the impact.

✦ Lead with the 'what', then the 'why'
State your answer clearly first, then provide justification. Interviewers lose interest if you meander before landing.

✦ Acknowledge trade-offs
No solution is perfect. Showing awareness of trade-offs (speed vs accuracy, bias vs variance) signals maturity.

✦ Connect to production
Mentioning real-world constraints — latency, scalability, monitoring — shows you think beyond Jupyter notebooks.

✦ Ask clarifying questions
Before solving a problem, confirm the objective, data available, and success metrics. It shows you think like a practitioner.