

Machine Learning Interview Q&A Handbook (200+ Questions)

Topic-Classified Answers with Practical Python Examples (Light-IDE Code Snippets)

Lamhot Siagian

February 2026

Contents

Topic Index	3
1 Foundations & Learning Paradigms	4
1.1 What is supervised learning?	4
1.2 What is unsupervised learning?	4
1.3 What is overfitting, and how can you prevent it?	4
1.4 What is regularization in machine learning?	5
1.5 What is bias-variance tradeoff?	5
1.6 What is feature selection?	5
1.7 How does Random Forest work?	5
1.8 What is Support Vector Machine (SVM)?	6
1.9 What are neural networks?	6
1.10 What is dropout in neural networks?	6
1.11 What is early stopping?	6
1.12 What is pruning in decision trees?	6
1.13 Give an example of transfer learning.	6
1.14 What is an autoencoder?	7
1.15 What is early stopping in neural networks?	7
1.16 What is dropout in neural networks?	7
1.17 What is the difference between supervised and unsupervised learning in	7
1.18 What is feature drift?	7
1.19 What is concept drift?	7
1.20 What is Semi-supervised Learning?	8
1.21 What is active learning?	8
1.22 What is label smoothing?	8
1.23 What is XGBoost and why is it so popular?	8
1.24 What is regularization (L1, L2) and how do they help in ML?	8
1.25 How does early stopping work and why does it improve generalization?	9
1.26 What is transfer learning? (Advanced context)	9
1.27 What is label propagation in semi-supervised learning?	10
1.28 What are variational autoencoders (VAEs)?	10
1.29 What is transfer learning "domain adaptation"?	10
1.30 Explain the difference between bagging and boosting in real-world use.	10
1.31 What is self-supervised learning?	10
1.32 What is active sampling in ML?	10
1.33 What is knowledge transfer in multi-task learning?	11
1.34 What is elastic net regularization?	11
1.35 What is polynomial regression?	11
1.36 What are the limitations of Deep Learning?	12
1.37 What is active learning and when is it useful?	12
1.38 What is batch normalization and why is it so critical for deep networks?	12
1.39 What is dropout and how does it help prevent overfitting?	12
1.40 What is early stopping?	13
1.41 What is transfer learning and why is it transformative for small datasets?	13
1.42 What is feature drift and concept drift?	13

2	Data Preparation & Feature Engineering	13
2.1	What is feature scaling, and why is it important?	13
2.2	Explain One-Hot Encoding.	14
2.3	What is data normalization?	14
2.4	How do you handle missing values in data?	14
2.5	What is feature engineering?	14
2.6	Give examples of feature engineering.	14
2.7	What is label encoding?	15
2.8	What is DBSCAN?	15
2.9	What is batch normalization?	15
2.10	What is data augmentation?	15
2.11	What is an outlier?	15
2.12	What is data pipeline?	15
2.13	What is data versioning and why is it important?	16
2.14	What is feature hashing?	16
2.15	What is cross-validation leakage and how do you avoid it?	16
2.16	How do you handle categorical variables in tree-based models as	16
2.17	What are positional encodings in Transformer models?	17
2.18	What is CatBoost and how does it handle categorical features?	17
2.19	What are Bayesian neural networks, and what's their advantage?	17
2.20	What is the Gumbel-softmax trick?	17
2.21	What is robust regression?	17
2.22	What is the role of feature importance in ML projects?	17
3	Model Evaluation, Validation & Experimentation	18
3.1	How do you evaluate a classification model?	18
3.2	Explain cross-validation.	18
3.3	What is hyperparameter tuning?	18
3.4	What is a confusion matrix?	19
3.5	What is the purpose of the ROC Curve?	19
3.6	What is precision and recall?	19
3.7	What is model deployment?	19
3.8	What is ensemble learning?	19
3.9	Explain Gradient Boosting.	19
3.10	What are Confusion Matrix terms: TP, FP, TN, FN?	20
3.11	What is the F1 score?	20
3.12	What are convolutional neural networks (CNNs)?	20
3.13	What is a confusion matrix used for?	20
3.14	What is silhouette score?	20
3.15	What is a ROC-AUC score?	20
3.16	What is precision-recall tradeoff?	21
3.17	What is pipeline in scikit-learn?	21
3.18	What is the Transformer architecture?	21
3.19	What is reinforcement learning?	21
3.20	What is Markov Decision Process (MDP)?	21
3.21	What is data leakage?	21
3.22	What is hyperparameter optimization?	21
3.23	What is precision-recall curve?	22

3.24	What is model calibration?	22
3.25	What is hyperparameter search space?	22
3.26	What is stratified k-fold cross-validation and when should you use it?	22
3.27	Explain the ROC curve, AUC, and their practical significance.	22
3.28	What is the difference between recall and specificity?	23
3.29	What is quantization in deploying ML models?	23
3.30	What is out-of-bag (OOB) error in random forests?	23
3.31	What is pipelining in MLOps?	24
3.32	What is data sharding?	24
3.33	What is evolutionary computation in ML?	24
3.34	What is multi-objective optimization?	24
3.35	What is transferability in adversarial ML?	24
3.36	What is hyperparameter tuning with Bayesian Optimization?	24
3.37	Why are ensemble models often used in Kaggle competitions?	25
3.38	What is class imbalance and its impact?	25
3.39	How do you address class imbalance?	25
3.40	What is model calibration and why is it important?	25
3.41	What is bootstrapping and why is it used?	26
4	Classical Supervised Algorithms	26
4.1	Explain the difference between classification and regression.	26
4.2	How does k-Nearest Neighbors (kNN) work?	26
4.3	What is a perceptron?	26
4.4	What is cross entropy loss?	27
4.5	What are decision trees and how do they work?	27
4.6	What is entropy in decision trees?	27
4.7	Explain the difference between parametric and non-parametric models.	27
4.8	What is the bias node in neural networks?	27
4.9	What is the softmax function?	28
4.10	What is multi-class classification?	28
4.11	What is multi-label classification?	28
4.12	What is time series forecasting?	28
4.13	What is ARIMA?	28
4.14	What is the difference between hard and soft classification?	28
4.15	What is multi-output regression?	29
4.16	What is Quantile Regression?	29
4.17	What is distance metric learning?	29
4.18	What is ensemble stacking and how does it work in practice?	29
4.19	What is LIME (Local Interpretable Model-agnostic Explanations)?	30
4.20	What is intercept bias in regression?	30
5	Ensembles & Boosting	30
5.1	What is the difference between bagging and boosting?	30
5.2	Name common ensemble algorithms.	30
5.3	What is stacking in ensembles?	31
5.4	What is bagging and give an example?	31
5.5	What is boosting and give an example?	31
5.6	How can you handle imbalanced datasets?	31

5.7	What is ensemble bagging vs. boosting in one line?	31
6	Unsupervised Learning & Clustering	32
6.1	What is k-means clustering?	32
6.2	What is hierarchical clustering?	32
6.3	What is anomaly detection?	32
6.4	What is the elbow method in clustering?	32
6.5	What is the silhouette coefficient formula?	32
7	Dimensionality Reduction	32
7.1	What is Principal Component Analysis (PCA)?	32
7.2	Explain dimensionality reduction.	33
7.3	What is t-SNE?	33
7.4	What is singular value decomposition (SVD)?	33
8	Deep Learning Core Concepts	33
8.1	Explain batch gradient descent.	33
8.2	What does the learning rate control?	33
8.3	What are recurrent neural networks (RNNs)?	34
8.4	What is the activation function in a neural network?	34
8.5	What is the ReLU activation function?	34
8.6	What is Xavier/Glorot initialization?	34
8.7	What is Gradient Vanishing/Explosion?	34
8.8	What is an epoch?	34
8.9	What is mini-batch gradient descent?	34
8.10	What is the main role of the loss function?	35
8.11	What is a generator in deep learning?	35
8.12	What are Generative Adversarial Networks (GANs)?	35
8.13	What is LSTM?	35
8.14	What is a learning rate scheduler?	35
8.15	What is attention mechanism in deep learning?	35
8.16	What is the exploding gradient problem?	35
8.17	What is a custom loss function?	36
8.18	What is the vanishing gradient problem mainly due to?	36
8.19	Explain learning rate annealing and why you might use it.	36
8.20	Describe bag-of-words vs. embeddings for NLP.	36
8.21	What is a graph neural network (GNN)?	36
8.22	What is Nash Equilibrium in ML?	37
8.23	What is hierarchical reinforcement learning?	37
9	NLP & Transformers	37
9.1	What are word embeddings?	37
9.2	What is word2vec?	37
9.3	What is the difference between Bag of Words and TF-IDF?	37
9.4	What is BERT?	38
9.5	What is negative sampling?	38
9.6	What is multi-head attention and why is it important in Transformers?	38
9.7	What is attention masking?	38

10 Reinforcement Learning & Decision Making	39
10.1 What is Q-learning?	39
10.2 What is reinforcement learning "reward shaping"?	39
10.3 What is Monte Carlo Tree Search?	39
10.4 What is bandit learning?	39
10.5 What is the difference between exploration and exploitation in RL?	39
11 MLOps, Deployment & Production	40
11.1 What is model serialization and why is it important?	40
11.2 Name tools for model deployment.	40
11.3 Why is interpretability increasingly important in ML?	40
12 Explainability, Robustness & Privacy	40
12.1 What is model interpretability and why is it important?	40
12.2 List model interpretability techniques.	40
12.3 What is explainable AI (XAI)?	41
12.4 What is feature importance?	41
12.5 What is SHAP and how does it differ from LIME?	41
12.6 What is explainability vs. interpretability?	41
12.7 What is adversarial training in deep learning?	41
12.8 What is federated learning?	42
12.9 What is privacy-preserving ML?	42
12.10 What is differential privacy?	42
12.11 What are SHAP values and how do they aid explainability?	42
13 Graphs & Advanced Topics	42
13.1 Explain knowledge distillation.	42
13.2 What is meta-learning ("learning to learn")?	43
13.3 What is curriculum learning?	43
13.4 What is knowledge graph embedding?	43
13.5 What is node2vec?	43
14 Other	43
14.1 What is Gini impurity?	43
14.2 What is class imbalance?	44
14.3 (Missing in source PDF)	44
14.4 What is fine-tuning in deep learning?	44
14.5 What is zero-shot learning?	44
14.6 What is SMOTE?	44
14.7 What is early fusion and late fusion in multimodal learning?	44
14.8 What are residual connections and why do they work?	44
14.9 What is the curse of dimensionality?	45
14.10 What is continual learning?	45

Topic Index

Topic	Questions	Count
Foundations & Learning Paradigms	1–3, 11–12, 15, 17–18, 21, 23, 29, 40, 52–53, 74, 85, 110–114, 118, 135–137, 144, 147–148, 151, 157, 163, 185–186, 188, 190–192, 194, 196–197, 201, 203	42
Data Preparation & Feature Engineering	6, 13, 25–26, 41–43, 59, 76, 84, 106, 121, 123, 129, 132, 140, 142, 153, 160, 167, 176, 193	22
Model Evaluation, Validation & Experimentation	7–10, 19–20, 30–31, 36, 47–48, 54, 56, 60, 62, 66, 90, 92, 98, 100, 103, 109, 116–117, 124, 133–134, 139, 149, 156, 158, 168, 177–180, 195, 199–200, 204–205	41
Classical Supervised Algorithms	4–5, 22, 27, 37, 39, 44, 69, 71, 95–96, 107–108, 119, 122, 125–126, 131, 154, 189	20
Ensembles & Boosting	16, 32–35, 50, 89	7
Unsupervised Learning & Clustering	57–58, 67, 70, 127	5
Dimensionality Reduction	14, 63–64, 175	4
Deep Learning Core Concepts	24, 28, 55, 72–73, 75, 77–83, 88, 91, 101–102, 128, 138, 145, 173, 181–182	23
NLP & Transformers	68, 86–87, 93, 130, 141, 143	7
Reinforcement Learning & Decision Making	99, 164–166, 183	5
MLOps, Deployment & Production	61, 65, 198	3
Explainability, Robustness & Privacy	45–46, 104–105, 155, 159, 161, 169–171, 202	11
Graphs & Advanced Topics	150, 152, 162, 172, 174	5
Other	38, 49, 51, 94, 97, 115, 120, 146, 184, 187	10

1 Foundations & Learning Paradigms

1.1 What is supervised learning?

Answer. Supervised learning is a type of machine learning where the model learns from labelled data. Each training example consists of input pairs and expected outputs (labels). The algorithms aim to map inputs to outputs. Example: Classification (e.g., spam detection), Regression (e.g., house price prediction). **Python (example).**

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
print(clf.score(X_test, y_test))
```

1.2 What is unsupervised learning?

Answer. Unsupervised learning deals with unlabeled data. The algorithm tries to find hidden patterns or groupings in data, commonly used for clustering or dimensionality reduction. **Python (example).**

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
iris = load_iris()
X = iris.data
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
print(kmeans.labels_)
```

1.3 What is overfitting, and how can you prevent it?

Answer. Overfitting happens when a model learns noise and details from the training data to the extent that it negatively impacts performance on new data. Prevention: Regularization, using more data, feature selection, cross-validation, early stopping (for neural networks), simplification of the model.

1.4 What is regularization in machine learning?

Answer. Regularization is a technique to reduce model complexity and prevent overfitting by adding a penalty on model parameters. Common types are L1 (Lasso) and L2 (Ridge) regularization. **Python (example).**

```
(Ridge Regression):  
python  
from sklearn.linear_model import Ridge  
ridge = Ridge(alpha=1.0)  
ridge.fit(X_train, y_train)  
print(ridge.score(X_test, y_test))
```

1.5 What is bias-variance tradeoff?

Answer. The bias-variance tradeoff is a balance between a model that is too simple (high bias, underfits) and one that is too complex (high variance, overfits). The goal is to find the optimal model with both low bias and variance.

1.6 What is feature selection?

Answer. Feature selection is the process of selecting the most important variables for model training to prevent overfitting, reduce computation, and improve interpretability. **Python (example).**

```
from sklearn.feature_selection import SelectKBest, f_classif  
selector = SelectKBest(score_func=f_classif, k=2)  
X_new = selector.fit_transform(X, y)
```

1.7 How does Random Forest work?

Answer. Random Forest is an ensemble method that constructs multiple decision trees on bootstrapped samples and averages their predictions to improve accuracy and reduce overfitting. **Python (example).**

```
from sklearn.ensemble import RandomForestClassifier  
rf = RandomForestClassifier(n_estimators=100)  
rf.fit(X_train, y_train)  
print(rf.score(X_test, y_test))
```

1.8 What is Support Vector Machine (SVM)?

Answer. SVM is a supervised learning algorithm for classification and regression, which finds the optimal hyperplane that maximizes the margin between different classes. **Python (example).**

```
from sklearn.svm import SVC
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
```

1.9 What are neural networks?

Answer. Neural networks are computational models inspired by the human brain, consisting of interconnected layers of nodes (neurons). Used for both supervised and unsupervised learning, especially powerful for complex tasks.

1.10 What is dropout in neural networks?

Answer. Dropout is a regularization technique for neural networks where a fraction of neurons are randomly set to zero during training, helping prevent overfitting. **Python (example).**

```
(Keras):
python
from keras.layers import Dropout
model.add(Dropout(0.5))
```

1.11 What is early stopping?

Answer. Early stopping ends the training of a model when performance on a validation set stops improving, preventing overfitting.

1.12 What is pruning in decision trees?

Answer. Pruning removes parts of a tree that do not provide additional predictive power to avoid overfitting.

1.13 Give an example of transfer learning.

Answer. Using a pre-trained VGG16 model (from ImageNet) for feature extraction in a custom image classification problem. **Python (example).**

```
(Keras):  
python  
from tensorflow.keras.applications import VGG16  
model = VGG16(weights='imagenet', include_top=False)
```

1.14 What is an autoencoder?

Answer. An autoencoder is an unsupervised neural network for learning efficient (compressed) data representations, typically for dimensionality reduction or denoising.

1.15 What is early stopping in neural networks?

Answer. A regularization strategy that halts training when validation performance stops improving, helping to prevent overfitting.

1.16 What is dropout in neural networks?

Answer. A regularization method where random neurons are set to zero during training to reduce overfitting and improve generalization.

1.17 What is the difference between supervised and unsupervised learning in

Answer. Supervised learning uses labeled data for training; unsupervised learning finds structure in unlabeled data.

1.18 What is feature drift?

Answer. Feature drift occurs when the statistical properties of input features change over time, potentially degrading model performance.

1.19 What is concept drift?

Answer. Concept drift happens when the statistical relationship between input and output changes over time, impacting model predictions.

1.20 What is Semi-supervised Learning?

Answer. A learning paradigm where the model is trained on a small amount of labeled data plus a large amount of unlabeled data.

1.21 What is active learning?

Answer. A technique where the model interactively queries the user or oracle to label new data points, improving learning efficiency.

1.22 What is label smoothing?

Answer. A regularization technique where target labels are adjusted (smoothed) to avoid overconfidence in models, common in deep learning.

1.23 What is XGBoost and why is it so popular?

Answer. XGBoost is an efficient, optimized implementation of gradient boosting, supporting parallelization, regularization, missing value handling, and more.

Why popular:

Consistently high accuracy

Flexibility with many hyperparameters

Handles large datasets, missing values

Widely used in data science competitions

Python (example).

```
import xgboost as xgb
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
```

1.24 What is regularization (L1, L2) and how do they help in ML?

Answer. L1 Regularization (Lasso): Adds absolute value of weights to loss, encouraging sparsity (drives less important weights to zero, useful for feature selection).

L2 Regularization (Ridge): Adds squared magnitude of weights to loss, discouraging large weights and thus reducing overfitting.

How it helps: Prevents overfitting, improves generalization, and can result in simpler, more interpretable models.

Python (example).

```
from sklearn.linear_model import Lasso, Ridge
lasso = Lasso(alpha=0.1)
ridge = Ridge(alpha=1.0)
```

1.25 How does early stopping work and why does it improve generalization?

Answer. Early stopping monitors model performance on a validation set during training.

Training is stopped when validation loss fails to improve for a set number of epochs (patience).

Why it's useful: Prevents overfitting by not allowing the model to "memorize" the training data. The model is selected at the point where it best generalizes.

Practical tip: Use with deep learning, boosting, and any iterative training process with clear validation metrics.

1.26 What is transfer learning? (Advanced context)

Answer. Transfer learning leverages knowledge from a pre-trained model (often on a massive dataset) to rapidly adapt to a new but related task:

Fine-tuning: Unfreeze some or all layers and re-train on smaller, task-specific data.

Feature extraction: Use pre-trained model as a fixed feature extractor by only training new layers.

In NLP and vision, transfer learning can achieve strong results with less data and compute.

1.27 What is label propagation in semi-supervised learning?

Answer. Label propagation assigns class labels to unlabeled points based on their similarity to labeled examples—using a graph-based approach to spread known labels through the dataset.

1.28 What are variational autoencoders (VAEs)?

Answer. VAEs are a type of generative unsupervised neural network. They encode input to a probabilistic latent space, then decode to reconstruct data—enabling new sample generation, interpolation, and probabilistic reasoning.

1.29 What is transfer learning "domain adaptation"?

Answer. Domain adaptation specifically refers to transferring knowledge between differing but related data distributions—adjusting representations/models to maintain performance as distributions shift between source (training) and target (deployment) data.

1.30 Explain the difference between bagging and boosting in real-world use.

Answer. Bagging (e.g., Random Forest): Reduces variance and helps with overfitting by averaging many independent models. Boosting (e.g., XGBoost): Reduces bias by building sequential models, each correcting its predecessor's errors, often delivering stronger results for structured/tabular data.

1.31 What is self-supervised learning?

Answer. Self-supervised learning leverages the data's own structure for supervision.

Models generate their own labels from input data—e.g., predicting masked words in a sentence (BERT) or the next video frame.

Applications: Pre-training models for NLP, vision, and audio to learn reusable representations.

1.32 What is active sampling in ML?

Answer. Active sampling (often part of active learning) is a data selection strategy where the model identifies data points whose labels would most improve the model if known.

Why it matters: In situations where manual labeling is costly or slow (medical images, legal documents), actively selecting "uncertain" or "borderline" samples leads to faster performance gains with less data.

Techniques: Uncertainty sampling, query-by-committee, expected model change, diversity sampling.

1.33 What is knowledge transfer in multi-task learning?

Answer. In multi-task learning (MTL), knowledge transfer occurs when learning one task improves performance in another by sharing representations or features.

Example: A neural network learns to detect both "cars" and "trucks" in images; learning the shape of wheels helps both.

Benefit: Better generalization, reduced risk of overfitting, data efficiency—especially when some tasks have less data.

1.34 What is elastic net regularization?

Answer. Elastic Net combines L1 (lasso) and L2 (ridge) penalties in regression:

L1: Drives some coefficients to zero (feature selection).

L2: Shrinks all coefficients to prevent overfitting.

When useful: When there are many correlated predictors; balances variable selection and model stability.

1.35 What is polynomial regression?

Answer. A type of regression where input variables are raised to integer powers to capture non-linear trends.

Example: Fitting a parabolic curve for data with quadratic relationships.

Risk: Higher degrees capture more complexity but can overfit; regularization or cross-validation is important.

1.36 What are the limitations of Deep Learning?

Answer. Data Intensive: Requires large labeled datasets.

Computationally Expensive: High training/inference costs (time, hardware).

Lack of Interpretability: Often yields "black box" predictions.

Sensitive to Adversarial Attacks: Vulnerable to small, crafted input changes.

Overfitting: Can memorize if not regularized properly or data is scarce.

Deployment Difficulties: Sometimes hard to update or control large models in production.

1.37 What is active learning and when is it useful?

Answer. Active learning is a setting where a model can interactively query a human annotator for labels on the most uncertain or informative samples.

When useful: When labeling data is expensive/scarc; to maximize learning efficiency (e.g., medical AI, rare events).

1.38 What is batch normalization and why is it so critical for deep networks?

Answer. Batch normalization normalizes intermediate activations in a neural network, stabilizing and accelerating training.

Benefits: Faster convergence, higher learning rates, regularization effect, less sensitivity to initialization.

1.39 What is dropout and how does it help prevent overfitting?

Answer. Dropout randomly disables a proportion of neurons during each training pass, forcing redundancy in representation and discouraging any single pathway's dominance.

Result: Significant boost in test-time generalization, especially in large networks.

1.40 What is early stopping?

Answer. Early stopping halts training when a monitored score (e.g., validation loss) stops improving for a set number of iterations ("patience").

Effect: Reduces risk of overfitting; preserves the weights of the model that best generalizes to unseen data.

1.41 What is transfer learning and why is it transformative for small datasets?

Answer. Transfer learning means using pre-trained models as starting points on new—but related—tasks.

Why important: Achieves high accuracy with less data and computation, democratizing advanced ML for small or domain-specific datasets.

1.42 What is feature drift and concept drift?

Answer. Feature drift: Changes in the distribution of predictor variables.

Concept drift: Changes in the mapping from input features to the target variable.

Impact: Reduces model performance over time; continuous monitoring and retraining are needed in production.

2 Data Preparation & Feature Engineering

2.1 What is feature scaling, and why is it important?

Answer. Feature scaling standardizes the range of independent variables, especially important for algorithms like kNN and SVM. Common methods: Standardization (z-score), Min-Max scaling. **Python (example).**

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

2.2 Explain One-Hot Encoding.

Answer. One-hot encoding converts categorical variables into binary vectors. Each category becomes a new column with values 0 or 1. **Python (example).**

```
import pandas as pd
df = pd.DataFrame({'color': ['red', 'blue', 'green']})
one_hot = pd.get_dummies(df['color'])
print(one_hot)
```

2.3 What is data normalization?

Answer. Normalization transforms features to have a 0 mean and unit variance, or to fit within a fixed range (e.g., 0 to 1). **Python (example).**

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_norm = scaler.fit_transform(X)
```

2.4 How do you handle missing values in data?

Answer. Common methods: removing rows, replacing (imputing) with mean/median/mode, forward/backward fill. **Python (example).**

```
import pandas as pd
df.fillna(df.mean(), inplace=True)
```

2.5 What is feature engineering?

Answer. Feature engineering is creating new features or modifying existing ones from raw data to improve model performance.

2.6 Give examples of feature engineering.

Answer. Scaling, log transforms, encoding categoricals, extracting date/time features, interactions (products/sums), polynomial features.

2.7 What is label encoding?

Answer. Label encoding converts categorical labels into numeric codes for model use. **Python (example).**

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

2.8 What is DBSCAN?

Answer. DBSCAN is a density-based clustering algorithm that groups together points closely packed and marks as outliers points lying alone in low-density regions.

2.9 What is batch normalization?

Answer. A method that normalizes layer inputs for each mini-batch during training, improving speed and stability. **Python (example).**

```
(Keras):
python
from keras.layers import BatchNormalization
model.add(BatchNormalization())
```

2.10 What is data augmentation?

Answer. A process of creating new training samples by modifying existing ones (e.g., rotation, flipping images) to improve model robustness and generalization.

2.11 What is an outlier?

Answer. A data point significantly different from others—can distort statistical analyses and model performance.

2.12 What is data pipeline?

Answer. A series of data processing steps (cleaning, feature engineering, modeling, deployment) organized in a sequence for reproducibility and automation.

2.13 What is data versioning and why is it important?

Answer. Tracking different versions of datasets used in ML experiments, enabling reproducibility and auditing of results.

2.14 What is feature hashing?

Answer. A feature engineering technique using a hash function to convert categorical variables into fixed-length vectors, often for NLP.

2.15 What is cross-validation leakage and how do you avoid it?

Answer. Cross-validation leakage (data leakage) occurs when information from outside the current training fold "leaks" into the fold, improperly inflating evaluation scores.

Example: Scaling or feature selection is performed before splitting into folds, allowing test data to influence the transformation.

How to avoid:

Always fit data transformations (scalers, encoders) inside the cross-validation loop or use scikit-learn's Pipeline to ensure transformations are applied only to the training fold.

Never peek at the test set during preprocessing or feature engineering.

2.16 How do you handle categorical variables in tree-based models as

Answer. Linear Models: Require numeric input, so you must perform label encoding or one-hot encoding.

Tree-based Models (e.g., Decision Trees, Random Forests): Can handle label-encoded categoricals natively, since splits occur on discrete values and the ordering is not learned.

Advanced: Some modern tree implementations (like CatBoost, LightGBM) have specialized handling for categorical variables without explicit

encoding.

2.17 What are positional encodings in Transformer models?

Answer. Since Transformers have no inherent notion of sequence (unlike RNNs), positional encodings inject information about order/position of tokens in a sequence—typically using sinusoidal or learned vectors added to input embeddings.

2.18 What is CatBoost and how does it handle categorical features?

Answer. CatBoost is a gradient boosting library designed to natively handle categorical variables without explicit encoding, using permutation-driven statistics during training, so you don't need to "one-hot" encode categoricals.

2.19 What are Bayesian neural networks, and what's their advantage?

Answer. Bayesian neural networks add uncertainty estimation to weights and predictions, enabling better calibrated confidence intervals, outlier detection, and robust decision making, especially useful in risk-sensitive applications like medicine and finance.

2.20 What is the Gumbel-softmax trick?

Answer. Enables differentiable sampling from a categorical distribution—crucial for training neural networks where discrete choices are involved, such as in neural architecture search or NLP models.

2.21 What is robust regression?

Answer. Regression techniques less sensitive to outliers compared to ordinary least squares (OLS).

Examples: RANSAC, Huber regression, Theil–Sen estimator.

2.22 What is the role of feature importance in ML projects?

Answer. Feature importance scores help identify which features most influence the predictions.

Why important: Guides data collection, feature engineering, and trust in

model decisions.

In practice: Used for model interpretation, debugging, compliance, and scientific discovery.

3 Model Evaluation, Validation & Experimentation

3.1 How do you evaluate a classification model?

Answer. Common metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC. Confusion Matrix is also used for detailed evaluation. **Python (example).**

```
from sklearn.metrics import classification_report, confusion_matrix
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

3.2 Explain cross-validation.

Answer. Cross-validation splits data into multiple folds; each fold is used as a validation set while the others are used for training. Most common is k-fold cross-validation. **Python (example).**

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, X, y, cv=5)
print(scores)
```

3.3 What is hyperparameter tuning?

Answer. Hyperparameter tuning involves finding the best combination of model settings (like tree depth, number of neighbors) to maximize performance. **Python (example).**

```
from sklearn.model_selection import GridSearchCV
params = {'max_depth': [3, 5, 7]}
grid = GridSearchCV(DecisionTreeClassifier(), params, cv=3)
grid.fit(X, y)
print(grid.best_params_)
```

3.4 What is a confusion matrix?

Answer. A confusion matrix summarizes the performance of a classification model by showing true positive, false positive, true negative, and false negative counts. **Python (example).**

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

3.5 What is the purpose of the ROC Curve?

Answer. The ROC Curve (Receiver Operating Characteristic) is a graphical plot showing the performance of a binary classifier as the discrimination threshold is varied. **Python (example).**

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
print(auc(fpr, tpr))
```

3.6 What is precision and recall?

Answer. Precision: Ratio of true positives to all predicted positives.

Recall: Ratio of true positives to all actual positives.

3.7 What is model deployment?

Answer. Model deployment is the process of integrating a trained model into a production environment where it can make real-time predictions on new data.

3.8 What is ensemble learning?

Answer. Ensemble learning combines multiple models (often weak learners) to improve overall model performance, accuracy, and robustness.

3.9 Explain Gradient Boosting.

Answer. Gradient Boosting builds models sequentially, each new one trained on the residuals (errors) of the previous model, which improves prediction accuracy. **Python (example).**

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
```

3.10 What are Confusion Matrix terms: TP, FP, TN, FN?

Answer. TP: True Positive

FP: False Positive

TN: True Negative

FN: False Negative

3.11 What is the F1 score?

Answer. F1 score is the harmonic mean of precision and recall, balancing both in a single metric.

3.12 What are convolutional neural networks (CNNs)?

Answer. CNNs are specialized neural networks for processing data with grid-like topology (e.g., images), using convolutional layers to automatically extract features.

3.13 What is a confusion matrix used for?

Answer. It's used to visually assess performance of a classification model by showing correct and incorrect predictions across classes.

3.14 What is silhouette score?

Answer. It evaluates how similar an object is to its own cluster versus other clusters, ranging from -1 to 1 (higher is better).

3.15 What is a ROC-AUC score?

Answer. Area Under the Receiver Operating Characteristic Curve; measures classifier's discrimination ability (higher is better).

3.16 What is precision-recall tradeoff?

Answer. It's the balance between precision (minimizing false positives) and recall (minimizing false negatives); often a model can't maximize both.

3.17 What is pipeline in scikit-learn?

Answer. A pipeline chains preprocessing steps and estimator(s) so the entire workflow can be treated as one composite model. **Python (example).**

```
from sklearn.pipeline import Pipeline
pipe = Pipeline([('scaler', StandardScaler()), ('clf', LogisticRegression())])
pipe.fit(X_train, y_train)
```

3.18 What is the Transformer architecture?

Answer. A deep learning model using self-attention to process sequences in parallel, highly effective for language tasks (e.g., GPT, BERT).

3.19 What is reinforcement learning?

Answer. Learning process where agents take actions in an environment to maximize a reward signal over time.

3.20 What is Markov Decision Process (MDP)?

Answer. A framework for modeling decision-making, involving states, actions, rewards, and transition probabilities, used in RL.

3.21 What is data leakage?

Answer. When information from outside the training dataset is used to create the model, leading to over-optimistic evaluations.

3.22 What is hyperparameter optimization?

Answer. Systematic approach to finding the best hyperparameters for a model, using Grid Search, Random Search, Bayesian Optimization.

3.23 What is precision-recall curve?

Answer. A plot that visualizes the trade-off between precision and recall for different probability thresholds of a classifier.

3.24 What is model calibration?

Answer. Adjusting the model's predicted probabilities to better reflect true likelihood, often with Platt scaling or isotonic regression.

3.25 What is hyperparameter search space?

Answer. Range and values of all tunable parameters considered during optimization.

3.26 What is stratified k-fold cross-validation and when should you use it?

Answer. Stratified k-fold keeps the class distribution consistent across all folds, ensuring each training/validation set is representative of the whole dataset.

When to use: For imbalanced classification tasks, you should prefer stratified k-fold so minority/majority classes are equally represented in each split.

Python (example).

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=5)
for train_idx, val_idx in skf.split(X, y):
    X_train, X_val = X[train_idx], X[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]
```

3.27 Explain the ROC curve, AUC, and their practical significance.

Answer. ROC Curve: Plots True Positive Rate (sensitivity) vs. False Positive Rate at various thresholds. Used to evaluate binary classifier performance across thresholds.

AUC (Area Under Curve): Summarizes the ROC curve with a single number (maximum = 1). Higher AUC means better discrimination between positive

and negative classes.

Practical significance: Particularly useful for imbalanced problems and when you care about rank ordering or threshold optimization, not just accuracy.

Python (example).

```
from sklearn.metrics import roc_curve, roc_auc_score
y_prob = clf.predict_proba(X_test)[:,-1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)
```

3.28 What is the difference between recall and specificity?

Answer. Recall (True Positive Rate): Measures the proportion of actual positives captured by the model.

$\text{Recall} = \frac{TP}{TP + FN}$

Specificity (True Negative Rate): Measures the proportion of actual negatives correctly identified.

$\text{Specificity} = \frac{TN}{TN + FP}$

Use cases: Use recall when missing positives is costly (medical diagnoses).

Use specificity when false positives are costly (spam filters).

3.29 What is quantization in deploying ML models?

Answer. Quantization reduces the precision of the numbers (weights/activations) used in a model (e.g., float32 to int8), making it efficient for edge devices/mobile deployment, with normally only minor loss in accuracy.

3.30 What is out-of-bag (OOB) error in random forests?

Answer. OOB error estimates generalization accuracy using, for each tree, the samples not included ("out-of-bag") in that tree's bootstrap sample. Random forest combines these OOB predictions as a built-in cross-validation metric—no need to hold out validation data.

3.31 What is pipelining in MLOps?

Answer. Pipelining in MLOps automates and chains all stages of ML (data collection, preprocessing, training, testing, deployment) to improve reproducibility, monitoring, and rapid iteration (e.g., using Kubeflow, MLflow, or Airflow).

3.32 What is data sharding?

Answer. Dividing large datasets into smaller, manageable pieces (shards) stored or processed on different machines.

Why: Improves scalability and performance for distributed/decentralized machine learning systems.

3.33 What is evolutionary computation in ML?

Answer. A family of optimization algorithms inspired by biological evolution (e.g., genetic algorithms). Used to optimize models, hyperparameters, or generate novel neural architectures.

3.34 What is multi-objective optimization?

Answer. Optimization involving more than one conflicting metric (e.g., accuracy and model size). Solutions are evaluated by Pareto optimality—improvement in one objective means tradeoff in another.

Example: Model accuracy vs. latency trade-offs in mobile deployment.

3.35 What is transferability in adversarial ML?

Answer. Adversarial examples generated for one model can often fool different models (even with varied architectures/training).

Implication: Model robustness should be evaluated against both "white-box" and "black-box" attackers.

3.36 What is hyperparameter tuning with Bayesian Optimization?

Answer. A search strategy that models the performance metric as a probabilistic function

of hyperparameters (e.g., Gaussian Process), choosing new hyperparameters to try based on uncertainty and expected improvement.

Pros: Needs fewer runs than grid/random search; often faster convergence.

3.37 Why are ensemble models often used in Kaggle competitions?

Answer. Ensembles combine strengths and compensate for weaknesses of various models, pushing accuracy and generalization higher than any single model.

Kaggle effect: Final "winning" solutions often average or stack dozens of diverse models for a performance edge.

3.38 What is class imbalance and its impact?

Answer. Imbalanced data means most samples belong to one class (majority), with very few in another (minority), such as fraud detection or rare disease diagnosis.

Impact: Models "cheat" by always predicting the majority class—minority detection suffers; misleading accuracy metrics.

3.39 How do you address class imbalance?

Answer. Rebalancing the Data: Oversample minority (SMOTE), undersample majority.

Adjusting Algorithms: Use class weights, modify decision thresholds.

Careful Evaluation: Focus on recall, precision, F1, ROC-AUC—never just overall accuracy.

3.40 What is model calibration and why is it important?

Answer. Calibration means predicted probabilities reflect real-world frequencies.

Example: In a calibrated medical model, 70% risk should mean the event actually happens 70% of the time.

Why important: Essential for risk-sensitive applications; can be achieved

with Platt scaling, isotonic regression.

3.41 What is bootstrapping and why is it used?

Answer. Bootstrapping involves generating many resampled datasets (with replacement) from the original data to estimate variability, build confidence intervals, or power bagging/ensembles.

Advantages: Provides insights into model stability and predictions; essential for statistical inference and robust modeling.

4 Classical Supervised Algorithms

4.1 Explain the difference between classification and regression.

Answer. A compact comparison is shown below.

Feature	Classification	Regression
Output type	Discrete categories	Continuous values
Example	Spam vs. ham email detection	House price prediction
Algorithm examples	Logistic Regression, SVM	Linear Regression, SVR

4.2 How does k-Nearest Neighbors (kNN) work?

Answer. kNN is a non-parametric algorithm that classifies a point by looking at the 'k' closest labeled data points in the feature space. For regression, it averages the values of k nearest neighbors.

Python (example).

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
```

4.3 What is a perceptron?

Answer. A perceptron is the simplest neural network, consisting of one layer with only one neuron, used for binary linear classification. **Python (example).**

```
from sklearn.linear_model import Perceptron
p = Perceptron()
p.fit(X_train, y_train)
```

4.4 What is cross entropy loss?

Answer. Cross entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. It increases as the predicted probability diverges from the actual label.

4.5 What are decision trees and how do they work?

Answer. Decision trees classify data by splitting nodes on feature values, choosing splits based on criteria like Gini impurity or entropy, until pure or max depth is reached.

4.6 What is entropy in decision trees?

Answer. Entropy measures the impurity or disorder in a node, used for determining optimal splits (as in ID3 algorithm).

4.7 Explain the difference between parametric and non-parametric models.

Answer. Key differences are summarized below.

Parametric	Non-Parametric
Assumes data distribution	Makes fewer assumptions
Fewer parameters	More flexible; often more parameters
Example: Linear Regression	Example: kNN, Decision Trees

4.8 What is the bias node in neural networks?

Answer. The bias node allows shifting the activation function curve and improves learning, similar to intercept in linear regression.

4.9 What is the softmax function?

Answer. Softmax converts a vector of raw scores (logits) into probabilities that sum to 1, commonly used in the output layer of multiclass classification neural networks. **Python (example).**

```
import numpy as np
def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum(axis=0)
```

4.10 What is multi-class classification?

Answer. Prediction where outputs can belong to more than two categories (e.g., classifying types of fruit).

4.11 What is multi-label classification?

Answer. A problem where each input can be assigned multiple labels (e.g., tagging images with multiple objects).

4.12 What is time series forecasting?

Answer. Predicting future values based on historical data with temporal sequence, using models like ARIMA, LSTM.

4.13 What is ARIMA?

Answer. Auto-Regressive Integrated Moving Average—a popular linear time series forecasting model.

4.14 What is the difference between hard and soft classification?

Answer. Key differences are summarized below.

Hard Classification	Soft Classification
Predicts class labels only	Predicts probabilities for each class

4.15 What is multi-output regression?

Answer. Predicting multiple continuous values simultaneously in a regression task.

4.16 What is Quantile Regression?

Answer. A regression technique predicting the conditional quantile values (rather than mean), useful for modeling median or ranges.

4.17 What is distance metric learning?

Answer. Learning an appropriate distance function (rather than using Euclidean, etc.) for tasks like clustering, retrieval.

4.18 What is ensemble stacking and how does it work in practice?

Answer. Stacking is an ensemble learning technique that combines multiple base models (like decision trees, SVMs, neural networks) by training a new model (meta-learner) on their outputs.

How it works:

1. Split the training data into two parts.
2. Train several base models on the first part.
3. Use these models to predict the second part ("out-of-fold" predictions).
4. Train a meta-model using these predictions as input features to learn optimal combinations.
5. At prediction time, all base models predict on the test data, and their outputs feed into the meta-model for the final prediction.

Python (example).

```
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
estimators = [
    ('tree', DecisionTreeClassifier()),
    ('svc', SVC(probability=True)),
```

```
]
clf = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegression())
clf.fit(X_train, y_train)
Use case: Stacking is popular in ML competitions (like Kaggle) to squeeze out
extra performance by leveraging strengths of diverse models.
```

4.19 What is LIME (Local Interpretable Model-agnostic Explanations)?

Answer. LIME explains model predictions by locally perturbing the input and fitting a simple interpretable model (like linear regression) on the prediction results to approximate the model's behavior in the local region.

4.20 What is intercept bias in regression?

Answer. The intercept (or bias) is the constant term in regression models, allowing the fitted function to move up or down to best match the data.

Without it: Predictions are forced through the origin, possibly distorting results.

5 Ensembles & Boosting

5.1 What is the difference between bagging and boosting?

Answer. Key differences are summarized below.

Bagging	Boosting
Trains models in parallel	Trains models sequentially
Aims to reduce variance	Aims to reduce bias
Example: Random Forest	Example: AdaBoost, XGBoost

5.2 Name common ensemble algorithms.

Answer. Random Forest, Bagging, Boosting (AdaBoost, Gradient Boosting, XGBoost), Stacking, Voting Classifier.

5.3 What is stacking in ensembles?

Answer. Stacking combines predictions from several base learners (with different strengths) using a meta-learner to give the final prediction.

5.4 What is bagging and give an example?

Answer. Bagging fits multiple models on random data subsets with replacement and averages results. Example: Random Forest.

5.5 What is boosting and give an example?

Answer. Boosting trains models sequentially, each focusing on correcting previous model errors. Example: AdaBoost, Gradient Boosting.

5.6 How can you handle imbalanced datasets?

Answer. Oversampling minority class (SMOTE)

Undersampling majority class

Use of class weights

Ensemble methods designed for imbalance

Python (example).

```
(class weights):  
python  
from sklearn.linear_model import LogisticRegression  
lr = LogisticRegression(class_weight='balanced')
```

5.7 What is ensemble bagging vs. boosting in one line?

Answer. Bagging averages predictions from many independent models; boosting builds models sequentially, each focusing on correcting predecessors' errors.

6 Unsupervised Learning & Clustering

6.1 What is k-means clustering?

Answer. K-means clustering partitions data into 'k' clusters by assigning points to nearest centroids iteratively. **Python (example).**

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)
```

6.2 What is hierarchical clustering?

Answer. A clustering method that builds a hierarchy of clusters either by successive merging (agglomerative) or splitting (divisive).

6.3 What is anomaly detection?

Answer. Anomaly detection aims to identify rare items, events, or observations that differ significantly from the majority of data.

6.4 What is the elbow method in clustering?

Answer. It's a technique to determine the optimal number of clusters in k-means by plotting the within-cluster sum of squares and looking for the "elbow" point where the decrease sharply slows.

6.5 What is the silhouette coefficient formula?

Answer. Given a point i : $s(i) = b(i) - a(i)$, $a(i) = \frac{1}{n(i)-1} \sum_{j \in C(i), j \neq i} d(i, j)$ where $a(i)$: average intra-cluster distance, $b(i)$: nearest-cluster distance.

7 Dimensionality Reduction

7.1 What is Principal Component Analysis (PCA)?

Answer. PCA is a dimensionality reduction technique that transforms data into fewer uncorrelated variables called principal components, preserving as much variance as possible. **Python (example).**

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
```

7.2 Explain dimensionality reduction.

Answer. Reducing the number of input variables in data (e.g., via PCA, t-SNE) to speed up training, aid visualization, and mitigate the curse of dimensionality.

7.3 What is t-SNE?

Answer. t-distributed Stochastic Neighbor Embedding (t-SNE) is an advanced dimensionality reduction technique mainly for high-dimensional data visualization.

7.4 What is singular value decomposition (SVD)?

Answer. A factorization method for real or complex matrices, decomposing them into the product of three matrices.

Uses: Dimensionality reduction (e.g., LSA for NLP), recommender systems, image compression.

8 Deep Learning Core Concepts

8.1 Explain batch gradient descent.

Answer. Batch gradient descent computes the gradient of the cost function using the entire training dataset and updates parameters in each iteration.

8.2 What does the learning rate control?

Answer. The learning rate controls the step size during gradient descent. Too high may cause divergence; too low, slow convergence.

8.3 What are recurrent neural networks (RNNs)?

Answer. RNNs are neural networks designed for sequential data (like time series, text), where outputs depend on previous computations (i.e., they have memory).

8.4 What is the activation function in a neural network?

Answer. An activation function introduces non-linearity, enabling the network to learn complex relationships. Examples: ReLU, sigmoid, tanh.

8.5 What is the ReLU activation function?

Answer. ReLU (Rectified Linear Unit) outputs zero for negative input and the input itself if positive: $f(x) = \max(0, x)$.

8.6 What is Xavier/Glorot initialization?

Answer. A technique for initializing neural network weights to prevent vanishing/exploding gradients, maintaining variance through layers.

8.7 What is Gradient Vanishing/Explosion?

Answer. When gradients become extremely small (vanishing) or large (exploding), hindering neural network training, especially in deep networks.

8.8 What is an epoch?

Answer. One epoch means one complete pass through the entire training dataset by the ML model.

8.9 What is mini-batch gradient descent?

Answer. An optimization algorithm that updates model parameters based on a small random subset (mini-batch) of data per iteration—balances speed and convergence.

8.10 What is the main role of the loss function?

Answer. The loss function quantifies how well predictions match actual target values, guiding weight updates during training.

8.11 What is a generator in deep learning?

Answer. A Python function or object that yields data batches on-the-fly, typically used for efficient data feeding during neural network training.

8.12 What are Generative Adversarial Networks (GANs)?

Answer. GANs consist of a generator and discriminator network competing against each other—the generator creates fake data, the discriminator distinguishes real from fake.

8.13 What is LSTM?

Answer. Long Short-Term Memory is an RNN architecture designed to capture long-range dependencies and mitigate vanishing gradients, used in sequences.

8.14 What is a learning rate scheduler?

Answer. A tool that adjusts the learning rate during training, often reducing it when improvement plateaus, for better convergence.

8.15 What is attention mechanism in deep learning?

Answer. A technique allowing neural networks to focus on specific parts of input data (such as tokens in NLP), improving performance in sequence tasks.

8.16 What is the exploding gradient problem?

Answer. When gradients grow uncontrollably, causing unstable updates during neural network training, especially in deep or recurrent architectures.

8.17 What is a custom loss function?

Answer. A user-defined metric for training models, implemented by subclassing or defining functions in libraries (Keras, PyTorch).

8.18 What is the vanishing gradient problem mainly due to?

Answer. Activation functions like sigmoid/tanh squash gradients, causing them to diminish in deep networks.

8.19 Explain learning rate annealing and why you might use it.

Answer. Learning rate annealing gradually decreases the learning rate during training, helping the model to converge smoothly and escape local minima.

Strategies: Step decay, exponential decay, reduce on plateau, etc.

Python (example).

```
(Keras):  
python  
from keras.callbacks import ReduceLROnPlateau  
ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5)  
Why use: Large initial rates speed up learning; smaller rates later refine the  
solution, avoiding oscillations.
```

8.20 Describe bag-of-words vs. embeddings for NLP.

Answer. Bag-of-words (BoW): Represents documents as unstructured collections (vectors) of word counts or indicators, ignoring order/context.

Embeddings: Map words/tokens to dense vectors in a continuous space, preserving semantic relationships and often capturing analogy or similarity (e.g., king-man+woman=queen).

Embeddings (Word2Vec, GloVe, BERT) are preferred for neural networks due to richer information.

8.21 What is a graph neural network (GNN)?

Answer. Neural networks that operate directly on graphs, learning representations for

nodes, edges, or whole graphs by aggregating information from a node's neighbors.

Popular for: Social network analysis, molecule property prediction, fraud detection.

8.22 What is Nash Equilibrium in ML?

Answer. In multi-agent settings (game theory), a Nash Equilibrium is a situation where no agent can benefit by changing its own strategy if others keep theirs unchanged.

Applications: Multi-agent reinforcement learning, GAN training dynamics.

8.23 What is hierarchical reinforcement learning?

Answer. Organizes learning into a hierarchy: high-level "managers" set subgoals or options that lower-level policies accomplish.

Why: Improves learning efficiency and scalability in complex, long-horizon tasks.

9 NLP & Transformers

9.1 What are word embeddings?

Answer. Word embeddings are dense vector representations of words (e.g., Word2Vec, GloVe) capturing semantic meaning for use in NLP.

9.2 What is word2vec?

Answer. An algorithm for learning word embeddings—maps words to a vector space such that similar words have close vector representations.

9.3 What is the difference between Bag of Words and TF-IDF?

Answer. Key differences are summarized below.

Bag of Words	TF-IDF
Counts word occurrences	Weighs words by frequency and distinctiveness
Ignores importance	Reflects word relevance in document/context

9.4 What is BERT?

Answer. Bidirectional Encoder Representations from Transformers—pre-trained NLP model using transformer architecture for understanding language context.

9.5 What is negative sampling?

Answer. A technique used in training word embeddings or recommendation systems, introducing fake (negative) examples to improve contrast with positive samples.

9.6 What is multi-head attention and why is it important in Transformers?

Answer. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions, enabling richer representation of input. For each "head," the model learns independent attention scores, then concatenates their outputs.

Importance:

Enables learning of multiple relationships simultaneously (syntax, semantics, etc.)

Boosts capability of networks for tasks like translation, summarization, and vision problems.

9.7 What is attention masking?

Answer. Attention masking is used in Transformers to avoid attending to certain positions, such as padding tokens or future tokens (for sequence-to-sequence modeling or autoregressive tasks). Ensures the model doesn't "cheat" by looking ahead.

10 Reinforcement Learning & Decision Making

10.1 What is Q-learning?

Answer. A reinforcement learning algorithm that learns optimal actions using the Q-value function for each state-action pair.

10.2 What is reinforcement learning "reward shaping"?

Answer. Reward shaping involves modifying the reward function to provide more frequent or informative feedback. This helps agents learn optimal behaviors faster or avoid undesirable states, but it must be done carefully to avoid unintended behaviors.

10.3 What is Monte Carlo Tree Search?

Answer. A search strategy for decision-making (especially in games), combining randomness (Monte Carlo simulation) and tree search to explore promising states efficiently. At each move, the algorithm simulates many games randomly to estimate the value of each possible choice, then follows the best path.
Famous use: AlphaGo's superhuman performance in the game of Go.

10.4 What is bandit learning?

Answer. Refers to multi-armed bandit problems where a learner balances "exploration" (trying new actions) and "exploitation" (choosing the best-known action) to maximize cumulative reward.
Applications: Online advertising, A/B testing, recommendation engines.

10.5 What is the difference between exploration and exploitation in RL?

Answer. Exploration: Trying new actions to discover rewards.
Exploitation: Leveraging known actions that give high rewards.
Balance: Key to ensure RL agents don't get stuck in suboptimal behaviors.

11 MLOps, Deployment & Production

11.1 What is model serialization and why is it important?

Answer. Serialization (saving) stores a trained model (e.g., using pickle or joblib in Python) so it can be reloaded without retraining. **Python (example).**

```
import joblib
joblib.dump(model, 'model.pkl') # Save
model = joblib.load('model.pkl') # Load
```

11.2 Name tools for model deployment.

Answer. Flask, FastAPI, Docker, TensorFlow Serving, AWS Lambda, Heroku, Google Cloud ML, Azure ML.

11.3 Why is interpretability increasingly important in ML?

Answer. As ML models are used for consequential decisions (loans, diagnostics, hiring), stakeholders must understand, trust, and audit model predictions.

Consequences: Legal, ethical, and practical—interpretability is at the heart of AI safety and responsible deployment.

12 Explainability, Robustness & Privacy

12.1 What is model interpretability and why is it important?

Answer. Model interpretability means how easily a human can understand the reasoning behind model decisions; important for trust, debugging, and compliance.

12.2 List model interpretability techniques.

Answer. Feature importance, SHAP values, LIME, Partial Dependence Plots.

12.3 What is explainable AI (XAI)?

Answer. A field focusing on making machine learning model predictions understandable by humans.

12.4 What is feature importance?

Answer. A measure of the contribution of each input variable to the prediction, often provided by tree-based models.

12.5 What is SHAP and how does it differ from LIME?

Answer. SHAP (SHapley Additive exPlanations) computes feature attributions for individual predictions based on cooperative game theory, providing consistent and theoretically-sound explanations; LIME is local and heuristic-based, while SHAP is global and has theoretical guarantees.

12.6 What is explainability vs. interpretability?

Answer. Interpretability: How well a human can understand the decisions of a model (simple models, feature importance).

Explainability: Tools/techniques to make "black box" models understandable, e.g., SHAP, LIME, counterfactual explanations.

12.7 What is adversarial training in deep learning?

Answer. Adversarial training is a method to make models robust against adversarial attacks—inputs that have been intentionally perturbed to fool a model. In practice, adversarial examples (generated using techniques like FGSM or PGD) are added to the training data, and the model is trained to classify both clean and adversarial samples correctly.

Use-case: Security-sensitive applications (e.g., autonomous vehicles, biometrics) to defend against malicious input manipulation.

12.8 What is federated learning?

Answer. A distributed ML approach where models are trained across multiple devices (e.g., smartphones), each with their own local data—without data leaving the device.

Benefits: Privacy, regulatory compliance, and less centralized data storage.

12.9 What is privacy-preserving ML?

Answer. Methods and protocols that allow useful models to be trained or deployed without exposing sensitive data. Includes techniques like differential privacy, homomorphic encryption, and secure multiparty computation.

Use-cases: Medical data, finance, federated learning.

12.10 What is differential privacy?

Answer. A mathematical guarantee that inclusion or exclusion of a single data point minimally affects the model's output. Achieved by adding calibrated noise to data or results, shielding individual records.

Industry adoption: Used by Apple, Google, and the U.S. Census Bureau.

12.11 What are SHAP values and how do they aid explainability?

Answer. SHAP (Shapley Additive exPlanations) quantify each feature's additive contribution to a model's prediction using cooperative game theory.

Strengths: Local and global interpretability; theoretical consistency; can be applied across many model types.

13 Graphs & Advanced Topics

13.1 Explain knowledge distillation.

Answer. A technique to transfer knowledge from a large, complex model ("teacher") to a smaller, faster "student" model by training the student to mimic the output or behavior of the teacher,

allowing for lightweight/efficient inference.

13.2 What is meta-learning ("learning to learn")?

Answer. Meta-learning algorithms learn how to adapt quickly to new tasks with minimal data by finding initialization or design strategies that generalize well—for example, MAML (Model-Agnostic Meta-Learning).

13.3 What is curriculum learning?

Answer. Curriculum learning is a training strategy where a model is first exposed to easier examples or subtasks, and the complexity gradually increases.

Why: Mimics how humans learn; can speed up convergence and improve generalization, especially in challenging deep learning tasks.

13.4 What is knowledge graph embedding?

Answer. A method to represent entities and relationships in a knowledge graph as low-dimensional vectors.

Why: Allows graphs to be used in ML models for applications like question answering, link prediction, and recommendation.

13.5 What is node2vec?

Answer. A scalable algorithm to generate vector representations (embeddings) of nodes in large networks. It uses biased random walks to capture diverse patterns of connectivity (community and structural equivalence).

14 Other

14.1 What is Gini impurity?

Answer. Gini impurity measures the probability of misclassifying a randomly chosen element if assigned randomly according to class distribution in a node.

14.2 What is class imbalance?

Answer. Class imbalance occurs when some classes are represented much more frequently than others in a dataset, making model evaluation and prediction harder.

14.3 (Missing in source PDF)

Answer. This question number is not present in the provided PDF; numbering jumps from Q50 to Q52.

14.4 What is fine-tuning in deep learning?

Answer. Adapting a pre-trained model on a subset of new data, useful for efficient training on specific tasks.

14.5 What is zero-shot learning?

Answer. Machine learning where the model can correctly predict on classes not seen in training, often using semantic relationships.

14.6 What is SMOTE?

Answer. Synthetic Minority Over-sampling Technique—generates synthetic examples for the minority class to address class imbalance. **Python (example).**

```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
X_res, y_res = sm.fit_resample(X, y)
```

14.7 What is early fusion and late fusion in multimodal learning?

Answer. Early fusion combines raw features from multiple modalities; late fusion combines predictions or learned representations.

14.8 What are residual connections and why do they work?

Answer. Residual (skip) connections add the input of a layer directly to its output (e.g., in ResNets):

$\text{output} = F(x) + x$

Why:

Allows gradients to flow more easily through deep networks

Eases optimization, enabling very deep neural architectures

14.9 What is the curse of dimensionality?

Answer. As the number of features increases:

Data becomes sparse (more combinations to cover).

Distance measures

14.10 What is continual learning?

Answer. Continual learning (lifelong learning) is the capability of models to learn new tasks sequentially while retaining knowledge learned from previous tasks (avoiding "catastrophic forgetting").

Importance: Real-world systems (e.g., personal assistants, robots) need to adapt over time without retraining from scratch.