scikit-learn - Interview -Questions -Answers

1. What is scikit-learn?

→ A Python library for machine learning with tools for classification, regression, clustering, and preprocessing.

2. Difference between supervised and unsupervised learning?

→ Supervised uses labeled data; unsupervised uses unlabeled data.

3. What is classification?

→ Predicting discrete labels (e.g., spam or not spam).

4. What is regression?

→ Predicting continuous numerical values (e.g., house price).

5. Common classification algorithms in scikit-learn?

→ Logistic Regression, KNN, SVM, Decision Trees, Random Forest.

6. Common regression algorithms?

→ Linear Regression, Ridge, Lasso, Decision Tree Regressor, Random Forest Regressor.

7. Difference between Logistic Regression and Linear Regression?

→ Logistic predicts probabilities/classes; Linear predicts continuous values.

8. What is train-test split?

→ Dividing dataset into training and testing sets using `train_test_split` .

9. How to scale features?

→ Using `StandardScaler` , `MinMaxScaler` , or `RobustScaler` .

10. What is cross-validation?

```
→ Splitting data into multiple folds to validate model performance.
```

11. Difference between K-Fold and Stratified K-Fold?

```
→ Stratified maintains class distribution across folds.
```

12. What is overfitting and underfitting?

→ Overfitting → model fits noise; Underfitting → model too simple.

13. How to prevent overfitting?

→ Regularization, pruning, feature selection, cross-validation.

14. What is regularization?

→ Technique to penalize large coefficients to reduce overfitting (L1, L2).

15. Difference between L1 and L2 regularization?

→ L1 → Lasso (sparse), L2 → Ridge (small coefficients).

16. How to evaluate classification models?

→ Accuracy, Precision, Recall, F1-score, ROC-AUC.

17. How to evaluate regression models?

→ MSE, RMSE, MAE, R^2 score.

18. What is confusion matrix?

→ Table showing true positives, true negatives, false positives, fals e negatives.

19. What is ROC curve?

→ Graph showing True Positive Rate vs False Positive Rate.

20. Difference between Bagging and Boosting?

→ Bagging → parallel ensemble (Random Forest), Boosting → sequential (XGBoost).

21. What is hyperparameter tuning?

→ Selecting best model parameters using GridSearchCV or RandomizedSea rchCV.

22. What is pipeline in scikit-learn?

→ Sequentially applying transformers and estimators to streamline wor kflow.

23. Difference between classifier's `predict()` and `predict_proba()` ?

→ `predict()` → classes, `predict_proba()` → probability of classes.

24. What is feature importance?

→ Measure of how much each feature contributes to model prediction.

25. Difference between Decision Tree and Random Forest?

→ Random Forest → ensemble of multiple trees for better performance a
nd stability.

26. Train-test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.2, random_state=42)
```

27. Train Logistic Regression classifier

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

28. Evaluate classification accuracy

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

29. Train Linear Regression model

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
```

30. Evaluate regression with R^2 score

```
from sklearn.metrics import r2_score
print(r2_score(y_test, y_pred))
```

31. Standardize features

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

32. Use KNN for classification

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

33. Decision Tree classifier

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth=3, random_state=42)
dt.fit(X_train, y_train)
```

34. Random Forest classifier

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

35. Compute confusion matrix

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

36. Compute classification report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

37. Cross-validation score

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

38. GridSearchCV example

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C':[0.1,1,10]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
```

39. RandomizedSearchCV example

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform
rand = RandomizedSearchCV(LogisticRegression(), param_distributions=
{'C':uniform()}, n_iter=10)
rand.fit(X_train, y_train)
```

40. Train Ridge regression

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
```

41. Train Lasso regression

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)
```

42. Predict probabilities in classifier

```
probs = model.predict_proba(X_test)
print(probs[:5])
```

43. Plot ROC curve

```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
plt.plot(fpr, tpr); plt.show()
```

44. Feature importance from Random Forest

```
importances = rf.feature_importances_
print(importances)
```

45. Train SVM classifier

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

46. Polynomial Features for regression

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
```

47. Train Gradient Boosting classifier

```
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier(n_estimators=100)
gb.fit(X_train, y_train)
```

48. Train Extra Trees classifier

```
from sklearn.ensemble import ExtraTreesClassifier
et = ExtraTreesClassifier(n_estimators=50)
et.fit(X_train, y_train)
```

49. Encode categorical features

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False)
X_encoded = encoder.fit_transform(X[['category_col']])
```

50. Pipeline example

In [1]:
```
from sklearn.preprocessing import RobustScaler
import numpy as np

X = np.array([[1], [2], [100], [4], [5]])
scaler = RobustScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled)
```

```
[[-1.        ]
 [-0.66666667]
 [32.        ]
 [ 0.        ]
 [ 0.33333333]]
```

🔹 Top ML Models – How They Work

1️⃣ Linear Regression

- Predicts a continuous target variable.
- Assumes a linear relationship: (y = w_1x_1 + w_2x_2 + ... + b)
- Finds the best-fit line by **minimizing Mean Squared Error (MSE)**.
- Uses **Ordinary Least Squares** to estimate coefficients.
- Sensitive to **outliers**.
- Can include **regularization** (Ridge/Lasso) to reduce overfitting.

2️⃣ Logistic Regression

- Predicts **binary outcomes** (0 or 1).
- Uses the **logistic (sigmoid) function**: (\sigma(z) = \frac{1}{1+e^{-z}})
- Estimates **probabilities** of classes.
- Loss function = **Log Loss / Cross-Entropy**.
- Can handle **multiclass** using one-vs-rest.
- Can include **regularization** to avoid overfitting.

3️⃣ Decision Tree

- Non-linear, tree-based model for classification/regression.
- Splits data based on features using **information gain (classification)** or **variance reduction (regression)**.
- Recursively partitions until stopping criteria (max depth, min samples).
- Easy to interpret and visualize.

- Prone to **overfitting**; needs pruning or limiting depth.

---

4 Random Forest

- Ensemble of **decision trees** (bagging technique).
- Each tree trained on a **random subset of data** and features.
- Final prediction = **majority vote (classification)** or **average (regression)**.
- Reduces overfitting and increases stability.
- Feature importance can be extracted.

---

5 Support Vector Machine (SVM)

- Finds the **hyperplane** that best separates classes.
- Maximizes **margin** between closest points (support vectors).
- Can use **kernel trick** for non-linear separation (RBF, polynomial).
- Works well in **high-dimensional spaces**.
- Sensitive to choice of **C** (regularization) and **gamma** (kernel parameter).

---

6 K-Nearest Neighbors (KNN)

- Instance-based model (lazy learning).
- Predicts class/value based on **majority/average of K nearest neighbors**.
- Distance metric (Euclidean, Manhattan) defines "closeness".
- No training phase; prediction can be slow for large datasets.
- Sensitive to **feature scaling**.

---

7 Gradient Boosting / XGBoost

- Ensemble method using **sequential trees**.
- Each tree corrects **errors of previous trees**.
- Optimizes **loss function** via gradient descent.
- Regularization helps reduce overfitting.
- Often produces high accuracy for tabular data.

---

8 Naive Bayes

- Probabilistic classifier based on **Bayes theorem**: (P(C|X) \propto P(X|C) P(C))
- Assumes **feature independence**.
- Works well for text classification (spam detection, sentiment analysis).
- Very fast and efficient.
- Handles categorical and continuous features (Gaussian NB).

---

9 Ridge & Lasso Regression

- **Ridge**: Linear regression + **L2 regularization** → penalizes large coefficients.

- **Lasso**: Linear regression + **L1 regularization** → can shrink some coefficients to 0 (feature selection).
- Helps reduce **overfitting**.
- Coefficients estimated using **gradient descent or closed-form solution**.
- Good for multicollinearity in features.

🔟 Principal Component Analysis (PCA)

- Dimensionality reduction technique.
- Finds **orthogonal directions (principal components)** capturing maximum variance.
- Projects data onto **lower-dimensional space**.
- Reduces noise and computation cost.
- Used as preprocessing for other ML models.

🔹 Assumptions of Top ML Models

1️⃣ Linear Regression

- Linear relationship between features and target.
- Residuals are normally distributed.
- Homoscedasticity: constant variance of residuals.
- No or little multicollinearity between features.
- Observations are independent.

2️⃣ Logistic Regression

- Logit (log-odds) of the outcome is linearly related to features.
- No multicollinearity among predictors.
- Observations are independent.
- Large sample size recommended for stable estimates.

3️⃣ Decision Tree

- Few assumptions about data distribution.
- Can handle non-linear relationships.
- Sensitive to irrelevant features and noise.
- Assumes features used for splitting are informative.

4️⃣ Random Forest

- Based on Decision Tree assumptions.
- Each tree assumes informative features in subsets.
- Trees are independent (bagging reduces correlation).
- Works well with large datasets and high-dimensional features.

---

5 Support Vector Machine (SVM)

- Data is somewhat separable in feature space (or transformable with kernel).
- Kernel choice impacts performance.
- Features should be scaled.
- Assumes independence of observations.

---

6 K-Nearest Neighbors (KNN)

- Assumes **similar points are close in feature space**.
- All features contribute equally to distance metric.
- Sensitive to irrelevant or scaled features.
- Requires meaningful distance metric.

---

7 Gradient Boosting / XGBoost

- Trees as weak learners assume informative splits exist.
- Observations are independent.
- Assumes residuals can be improved sequentially.
- Requires careful tuning to avoid overfitting.

---

8 Naive Bayes

- Assumes **conditional independence** of features given the class.
- Observations are independent.
- Features can follow specific distributions depending on variant (Gaussian, Multinomial, Bernoulli).

---

9 Ridge Regression

- Same as Linear Regression.
- Assumes linear relationship and residuals have constant variance.
- Penalizes large coefficients to reduce multicollinearity impact.

---

10 Lasso Regression

- Same as Linear Regression.
- Assumes linear relationship and independence of observations.
- Can perform feature selection if some coefficients shrink to zero.

---

11 Principal Component Analysis (PCA)

- Assumes linear relationship between variables.
- Variance captures the most important information.
- Features should be standardized if scales differ.

- Components are orthogonal (uncorrelated)

🔹 Underfitting vs Overfitting vs Bias vs Variance

Concept	Description	Causes / Notes	Effect on Model Performance
Underfitting	Model is too simple to capture underlying patterns in data	Too few features, low model complexity, insufficient training	High training error, high testing error
Overfitting	Model learns noise along with patterns; too complex for training data	Too many features, high model complexity, insufficient regularization	Low training error, high testing error
Bias	Error due to overly simplistic assumptions about data	Model cannot capture underlying patterns	High training error, high testing error
Variance	Error due to sensitivity to small fluctuations in training data	Model is too complex, learns noise	Low training error, high testing error

🔹 Quick Notes

- **High Bias → Underfitting**
- **High Variance → Overfitting**
- Goal: **Low Bias + Low Variance** (Balanced model)
- Use **cross-validation, regularization, feature selection** to manage bias-variance tradeoff.