Scikit_learn_interview _Questions

Phani Rajendra

Scikit_learn_interview_Questions

1. What is Scikit-Learn?

Answer: Scikit-Learn is an open-source machine learning library in Python that provides simple and efficient tools for data mining and analysis. It is built on top of NumPy, SciPy, and Matplotlib.

```
from sklearn import datasets
iris = datasets.load_iris()
print(iris.keys())
```

2. How do you install Scikit-Learn?

Answer: Scikit-Learn can be installed using pip:

Answer: Scikit-Learn can be installed using pip:

```
pip install scikit-learn
```

3. What are the key features of Scikit-Learn?

Answer:

- Simple and efficient tools for predictive data analysis.

- Built-in datasets for practice.

- Supports various supervised and unsupervised learning algorithms.

- Cross-validation and hyperparameter tuning tools.

4. What is a Pipeline in Scikit-Learn?

Answer: A `Pipeline` is used to streamline data processing by chaining multiple preprocessing steps and estimators.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', SVC())
])
```

5. How do you perform cross-validation in Scikit-Learn?

Answer: Cross-validation is performed using `cross_val_score()

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

iris = load_iris()
model = RandomForestClassifier()
scores = cross_val_score(model, iris.data, iris.target, cv=5)
print(scores)
```

6. Explain the train-test split function in Scikit-Learn ?

Answer: train_test_split is used to split a dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, te
print(X_train.shape, X_test.shape)
```

from sklearn.model_selection import train_test_split

from sklearn.datasets import load_iris

iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape)

7. What is feature scaling in Scikit-Learn?

Answer: Feature scaling ensures that all features have the same scale. Methods include:

- Standardization (StandardScaler)
- Min-Max Scaling (MinMaxScaler)

```
from sklearn.preprocessing import StandardScaler
import numpy as np

X = np.array([[10, 20], [30, 40], [50, 60]])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled)
```

8. What are the different types of machine learning algorithms available in Scikit-Learn?

Answer:

Supervised Learning: Regression (LinearRegression), Classification (SVC, RandomForestClassifier).
Unsupervised Learning: Clustering (KMeans), Dimensionality Reduction (PCA).
Reinforcement Learning: Not directly supported.

9. What are the types of datasets available in Scikit-Learn?

Answer: Scikit-Learn provides the following datasets:

- **Toy datasets:** load_iris(), load_digits(), load_wine().
- **Real-world datasets:** fetch_20newsgroups(), fetch_olivetti_faces().
- **Generated datasets:** make_classification(), make_regression().

```
from sklearn.datasets import load_iris
iris = load_iris()
print(iris.data.shape)
```

10. what are main features of Scikit-learn ?

Scikit-learn is a popular open-source machine learning library for Python. It provides a wide range of supervised and unsupervised learning algorithms through a consistent interface.

Main features of scikit-learn include:

• Simple and efficient tools for data mining and data analysis

• Accessible to everybody and reusable in various contexts

• Built on NumPy, SciPy, and matplotlib

• Open source, commercially usable - BSD license

Scikit-learn includes various modules for:

• Classification

• Regression

• Clustering

• Dimensionality reduction

• Model selection

• Preprocessing

11. How do you handle missing values in scikit-learn?

Scikit-learn provides the `SimpleImputer` class for handling missing values. It can replace missing values with a specified strategy such as mean, median, most frequent, or a constant value.

```
import numpy as np
from sklearn.impute import SimpleImputer

# Sample data with missing values
X = np.array([[1, 2, np.nan], [3, np.nan, 0], [np.nan, 4, 5]])

# Create imputer
imputer = SimpleImputer(strategy='mean')

# Fit and transform the data
X_imputed = imputer.fit_transform(X)

print("Original data:")
print(X)
print("\nImputed data:")
print(X_imputed)
```

12. What is the difference between fit(), transform(), and fit_transform() methods?

Answer:

• fit(): Calculates parameters from the data

• transform(): Applies the calculated parameters to transform the data

• fit_transform(): Combines fit() and transform() in one step

13. How do you perform one-hot encoding in scikit-learn?

Answer: Use OneHotEncoder from sklearn.preprocessing.

14. How do you save and load a trained model in scikit-learn?

Answer: pickle to save and load models.

15. What is the difference between classification and regression in scikit-learn?

Answer: Classification predicts discrete class labels, while regression predicts continuous values.

16. What is the purpose of the confusion_matrix function?

Answer: It computes the confusion matrix to evaluate classification accuracy.

```
from sklearn.metrics import confusion_matrix

y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
```