Scikit-Learn Cheatsheet

October 6, 2024

1 Introduction to Scikit-Learn

Scikit-Learn is a powerful Python library for machine learning, that facilitates preprocessing, model training, evaluation, and more.

1.1 Installation

```
pip install scikit-learn
```

1.2 Importing Scikit-Learn

```
import sklearn
from sklearn import datasets, model_selection,
    metrics
```

2 Data Preprocessing

2.1 Scaling Features

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

2.2 Encoding Categorical Features

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(
    categorical_data)
```

2.3 Splitting the Dataset

```
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)
```

3 Model Selection

3.1 K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

3.2 Decision Trees

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(max_depth=5)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)
```

3.3 Random Forest

```
from sklearn.ensemble import
    RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

4 Model Evaluation

4.1 Confusion Matrix

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
```

4.2 Classification Report

```
from sklearn.metrics import classification_report

report = classification_report(y_test, y_pred)
```

4.3 Cross-Validation

```
from sklearn.model_selection import
    cross_val_score

scores = cross_val_score(model, X, y, cv=5)
```

5 Unsupervised Learning

5.1 K-Means Clustering

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
```

5.2 Principal Component Analysis (PCA)

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data)
```

6 Hyperparameter Tuning

6.1 Grid Search

```
from sklearn.model_selection import GridSearchCV

params = {'n_neighbors': [3, 5, 7]}
grid_search = GridSearchCV(knn, param_grid=params)
grid_search.fit(X_train, y_train)
```

6.2 Randomized Search

```
from sklearn.model_selection import
    RandomizedSearchCV

random_search = RandomizedSearchCV(rf,
    param_distributions=params, n_iter=10)
random_search.fit(X_train, y_train)
```

7 Model Persistence

7.1 Saving a Model

```
import joblib

joblib.dump(knn, 'knn_model.pkl')
```

7.2 Loading a Model

```
knn = joblib.load('knn_model.pkl')
```