Python Libraries : Scikit-Learn

Agenda

- What is Scikit Learn Library?
- Scikit Learn
  - Datasets
  - Training the model
  - Data processing techniques
- Exploratory Data Analysis
- Imputation methods
- Scaling techniques
- One hot encoding

Introduction to Scikit Learn

- Scikit-learn is probably the most useful library for machine learning in Python.
- Sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.
- The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

  - **NumPy**: Base n-dimensional array package
  - **Pandas**: Data structures and analysis
  - **Matplotlib**: Comprehensive 2D/3D plotting
  - **IPython**: Enhanced interactive console
  - **Sympy**: Symbolic mathematics
  - **SciPy**: Fundamental library for scientific comp

Dataset Loading

INTERNSHIPSTUDIO

- A collection of data is called dataset. It is having the following two components –
  - **Features** – The variables of data are called its features. They are also known as predictors, inputs or attributes.
  - **Response** – It is the target variable that basically depends upon the feature variables. They are also known as label or output.
- Scikit-learn have few example datasets like iris and digits for classification and the Boston house prices for regression.

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

scikit learn

Toy datasets

scikit-learn comes with a few small standard datasets & can be loaded.
These are too small to be representative of real world machine learning tasks.

load_boston(*[, return_X_y])	Load and return the Boston house-prices dataset (regression).
load_iris(*[, return_X_y, as_frame])	Load and return the iris dataset (classification).
load_diabetes(*[, return_X_y, as_frame])	Load and return the diabetes dataset (regression).
load_digits(*[, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
load_linnerud(*[, return_X_y, as_frame])	Load and return the physical excercise linnerud dataset.
load_wine(*[, return_X_y, as_frame])	Load and return the wine dataset (classification).
load_breast_cancer(*[, return_X_y, as_frame])	Load and return the breast cancer Wisconsin dataset (classification).

Loading Iris Dataset

Example

Following is an example to load **iris** dataset −

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])
```

Output

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']
First 10 rows of X:
[
   [5.1 3.5 1.4 0.2]
   [4.9 3. 1.4 0.2]
   [4.7 3.2 1.3 0.2]
   [4.6 3.1 1.5 0.2]
   [5. 3.6 1.4 0.2]
   [5.4 3.9 1.7 0.4]
   [4.6 3.4 1.4 0.3]
   [5. 3.4 1.5 0.2]
   [4.4 2.9 1.4 0.2]
   [4.9 3.1 1.5 0.1]
]
```

Splitting the dataset

- To check the accuracy of our model, we can split the dataset into 2 pieces-**a training set** & **a testing set**.
-
- Use the training set to train the model and testing set to test the model. After that, we can evaluate how well our model did.

Example
The following example will split the data into 70:30 ratio, i.e. 70% data will be used as training data and 30% will be used as testing data.

```
from sklearn.model_selection import
train_test_split
X_train, X_test, y_train, y_test =
train_test_split( X, y, test_size = 0.3)
```

Splitting the Dataset

Example

The following example will split the data into 70:30 ratio, i.e. 70% data will be used as training data and 30% will be used as testing data. The dataset is iris dataset as in above example.

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

X = iris.data
y = iris.target

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size = 0.3, random_state = 1
)

print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)
```

Output

```
(105, 4)
(45, 4)
(105,)
(45,)
```

Train the Model

Next, we can use our dataset to train some prediction-model. As discussed, scikit-learn has wide range of **Machine Learning (ML) algorithms** which have a consistent interface for fitting, predicting accuracy, recall etc.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
classifier_knn = KNeighborsClassifier(n_neighbors =
3) classifier_knn.fit(X_train, y_train)
y_pred = classifier_knn.predict(X_test)
print("Accuracy:", metrics.accuracy_score(y_test,
y_pred))
```

Output

```
Accuracy: 0.9833333333333333
Predictions: ['versicolor', 'virgin
```

Preprocessing the Data

- As we are dealing with lots of data and that data is in raw form, before inputting that data to machine learning algorithms, we need to convert it into meaningful data. This process is called preprocessing the data.
- Scikit-learn has package named **preprocessing** for this purpose. The **preprocessing** package has many techniques-

- Binarisation
- Mean Removal
- Scaling
- Normalisation
  1. L1 Normalisation
  2. L2 Normalisation

Preprocessing techniques

INTERNSHIPSTUDIO

Binarisation

- This preprocessing technique is used when we need to convert our numerical values into Boolean values.
- In this example, we used **threshold value** = 0.5 hense all the values above 0.5 would be converted to 1, and all the values below 0.5 would be converted to 0.

```
import numpy as np
from sklearn import preprocessing
Input_data = np.array(
   [2.1, -1.9, 5.5],
   [-1.5, 2.4, 3.5],
   [0.5, -7.9, 5.6],
   [5.9, 2.3, -5.8]]
)
data_binarized = preprocessing.Binarizer(threshold=0.5).transform(input_data)
print("\nBinarized data:\n", data_binarized)
```

Output

```
Binarized data:
[
    [ 1.  0.  1.]
    [ 0.  1.  1.]
    [ 0.  0.  1.]
    [ 1.  1.  0.]
]
```

scikit learn

Preprocessing techniques

INTERNSHIPSTUDIO

Mean Removal

- This technique is used to eliminate the mean from feature vector so that every feature centered on zero.

```
#displaying the mean and the standard deviation of the input data
print("Mean =", input_data.mean(axis=0))
print("Stddeviation = ", input_data.std(axis=0))
#Removing the mean and the standard deviation of the input data

data_scaled = preprocessing.scale(input_data)
print("Mean_removed =", data_scaled.mean(axis=0))
print("Stddeviation_removed =", data_scaled.std(axis=0))
```

Output

```
Mean = [ 1.75 -1.275 2.2 ]
Stddeviation = [ 2.71431391 4.20022321 4.69414529]
Mean_removed = [ 1.11022302e-16 0.00000000e+00 0.00000000e+00]
Stddeviation_removed = [ 1. 1. 1.]
```

scikit learn

Preprocessing techniques

INTERNSHIPSTUDIO

Scaling

- We use this preprocessing technique for scaling the feature vectors. Scaling of feature vectors is important, because the features should not be synthetically large or small.

```
data_scaler_minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
data_scaled_minmax = data_scaler_minmax.fit_transform(input_data)
print ("\nMin max scaled data:\n", data_scaled_minmax)
```

Output

```
Min max scaled data:
[
   [ 0.48648649 0.58252427 0.99122807]
   [ 0. 1. 0.81578947]
   [ 0.27027027 0. 1. ]
   [ 1. 0.99029126 0. ]
]
```

scikit learn

Preprocessing techniques

Normalisation

- We use this preprocessing technique for modifying the feature vectors.
- Normalisation of feature vectors is necessary so that the feature vectors can be measured at common scale. There are two types of normalization-

L1 Normalisation

– It is also called Least Absolute Deviations. It modifies the value in such a manner that the sum of the absolute values remains always up to 1 in each row.

L2 Normalisation

– Also called Least Squares. It modifies the value in such a manner that the sum of the squares remains always up to 1 in each row.

INTERNSHIPSTUDIO

Q.1 What is Scikit-Learn?

Q.2 What is Data Preprocessing?

Q.3 What is Scaling?

Q.4 What is Binarisation?

Q.5 What is Normalisation?

Q.6 What is the purpose of Data Scaling?

scikit learn