

Introduction

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python
- It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering etc...
- It is built upon NumPy, SciPy and Matplotlib.

Installation

pip install scikit-learn

```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packag
```

```
import sklearn as sk
print(sk.__version__)
```

```
1.0.2
```

Load sklearn dataset

```
from sklearn.datasets import load_breast_cancer
br_canc = load_breast_cancer()
X = br_canc.data
y = br_canc.target
print(X.shape)
print(y.shape)
```

```
(569, 30)
(569,)
```

```
feature_names = br_canc.feature_names
target_names = br_canc.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
```

```
Feature names: ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
'mean smoothness' 'mean compactness' 'mean concavity'
'mean concave points' 'mean symmetry' 'mean fractal dimension'
'radius error' 'texture error' 'perimeter error' 'area error'
'smoothness error' 'compactness error' 'concavity error'
'concave points error' 'symmetry error' 'fractal dimension error'
'worst radius' 'worst texture' 'worst perimeter' 'worst area'
'worst smoothness' 'worst compactness' 'worst concavity']
```

```
'worst concave points' 'worst symmetry' 'worst fractal dimension']
Target names: ['malignant' 'benign']
```

```
import pandas as pd
df = pd.DataFrame(br_canc.data, columns=br_canc.feature_names)
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	



```
df["mean radius"].describe()
```

```
count    569.000000
mean      14.127292
std        3.524049
min        6.981000
25%       11.700000
50%       13.370000
75%       15.780000
max       28.110000
Name: mean radius, dtype: float64
```

Training and Testing Dataset Generation

```
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 = 1
)

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

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

(455, 30)
(114, 30)
(455,)
(114,)
```

Data Preprocessing

```
import numpy as np
data = [[21,45000,"govt"],[33,42000,"private"],[18,30000,"semi-govt"],[36,53000,"private"],
mydata = pd.DataFrame(data,columns=["age","salary","job"])
print(mydata)
```

	age	salary	job
0	21	45000	govt
1	33	42000	private
2	18	30000	semi-govt
3	36	53000	private
4	45	55000	govt
5	34	48000	semi-govt

Data Normalization: Min Max Method

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
numdata=mydata.iloc[:,0:2]
scaler_num=scaler.fit_transform(numdata)
scaler_num
```

```
array([[0.11111111, 0.6       ],
       [0.55555556, 0.48      ],
       [0.         , 0.        ],
       [0.66666667, 0.92      ],
       [1.         , 1.        ],
       [0.59259259, 0.72      ]])
```

Data Normalization: Standard Sclaer (Z-Score)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
numdata=mydata.iloc[:,0:2]
scaler_num=scaler.fit_transform(numdata)
scaler_num
```

```
array([[ -1.11056039, -0.06082053],
       [ 0.20026499, -0.42574371],
       [-1.43826674, -1.88543643],
       [ 0.52797133,  0.91230795],
       [ 1.51109037,  1.15559007],
       [ 0.30950044,  0.30410265]])
```

LabelEncoder

Encode target labels with value between 0 and n_classes-1.

This transformer should be used to encode target values

```

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
catdata=mydata['job']
data_encoded=le.fit_transform(catdata)
print(data_encoded)
print(le.classes_)

```

```

[0 1 2 1 0 2]
['govt' 'private' 'semi-govt']

```


Simple Linear Regression

```

data=pd.read_csv("/content/drive/MyDrive/AIT322-ML/Linear_Simple_Salary_Data.csv")
print(data.shape)
data.head()

```

```
(28, 2)
```

	YearsExperience	Salary	
0	1.1	39343	
1	1.3	46205	
2	1.5	37731	
3	2.0	43525	
4	2.2	39891	

```

from sklearn.linear_model import LinearRegression
X=data.drop("Salary", axis="columns")
y=data.Salary
model=LinearRegression()
model.fit(X,y)

```

```
LinearRegression()
```

```
model.score(X,y)
```

```
0.9613060196532883
```

```
model.predict([[7.5]])
```

```

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not
"X does not have valid feature names, but"
array([96687.85905288])

```

```

print("Coefficient:=",model.coef_)
print("Intercept:=",model.intercept_)
print("Formula: Y="+model.coef_+"*X "+model.intercept_)

```

```
print('Formula: Y = ',model.coef_[0], 'X + ',model.intercept_,
```

```
Coefficient:= [9490.86597718]
```

```
Intercept:= 25506.3642240701
```

```
Formula: Y= [9490.86597718] *X + 25506.3642240701
```

```
ans=9490.86597718*7.5+25506.3642240701
```

```
print(ans)
```

```
96687.8590529201
```

✓ 0s completed at 12:17 PM

