Supervised-Learning_flow-Copy1 (1)

August 19, 2022

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: from sklearn.datasets import load_boston
```

```
[3]: Boston_df=load_boston()
```

```
[4]: #Input Data
     data=pd.DataFrame(Boston_df.data,columns=Boston_df.feature_names)
```

```
[5]: #data = pd.read_csv('Advertising.csv', index_col = 0)
     data.head()
```

```
[5]:       CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
     0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
     1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
     2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
     3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
     4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

        PTRATIO       B  LSTAT
     0     15.3  396.90   4.98
     1     17.8  396.90   9.14
     2     17.8  392.83   4.03
     3     18.7  394.63   2.94
     4     18.7  396.90   5.33
```

```
[26]: target=pd.DataFrame(Boston_df.target,columns=['Target'])
```

```
[27]: target.head()
```

```
[27]:    Target
      0    24.0
      1    21.6
      2    34.7
      3    33.4
```

```
4    36.2
```

```
[6]: data.describe()
```

```
[6]:              CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            LSTAT
count  506.000000
mean    12.653063
std      7.141062
min      1.730000
25%      6.950000
50%     11.360000
75%     16.955000
max     37.970000
```

```
[7]: data.shape
```

```
[7]: (506, 13)
```

Add some EDA steps like scatterplot, pairplot, distplot, heatmap, outlier plot etc

0.0.1 Shuffle and Split

```
[8]: from sklearn.model_selection import train_test_split
```

[28]:

```
features = data[['CRIM']]
target = target[['Target']]
```

[29]:

```
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪train_size = .85, random_state=10)

print('Train set of features: ', X_train.shape)
print('Test set of features: ', X_test.shape)
print('Target for train: ', y_train.shape)
print('Target for test: ', y_test.shape)
```

```
Train set of features:  (430, 1)
Test set of features:  (76, 1)
Target for train:  (430, 1)
Target for test:  (76, 1)
```

[30]:

```
X_train
```

[30]:

```
         CRIM
225   0.52693
385  16.81180
60    0.14932
190   0.09068
6     0.08829
..        …
320   0.16760
15    0.62739
484   2.37857
125   0.16902
265   0.76162

[430 rows x 1 columns]
```

0.0.2 Learn the model on train data

[31]:

```
from sklearn.linear_model import LinearRegression
```

[32]:

```
my_model = LinearRegression()
```

[33]:

```
# Train the model
my_model.fit(X_train, y_train)
```

[33]:

```
LinearRegression()
```

[34]:

```
#B1 and B0 in the equation y = B0 + B1.x
my_model.coef_, my_model.intercept_
```

[34]:
```
(array([[-0.38165599]]), array([23.32281736]))
```

0.0.3 Test the model

[35]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

[36]:
```
y_pred = my_model.predict(X_test)
```

[37]:
```
# MAE

mean_absolute_error(y_pred, y_test)
```

[37]:
```
7.696726797496113
```

[38]:
```
# Compare with the true values, MSE

mean_squared_error(y_pred, y_test)
```

[38]:
```
116.94183429682572
```

[39]:
```
#RMSE
np.sqrt(mean_squared_error(y_pred, y_test))
```

[39]:
```
10.813964781560264
```

[21]:
```
import numpy as np
np.random.seed(10)
```

[22]:
```
np.random.randint(2, 10)
```

[22]:
```
3
```

[ ]:
```

```