

```
In [1]: import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.linear_model import SGDClassifier

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

CRIM: Per capita crime rate by town

ZN: Proportion of residential land zoned for lots over 25,000 sq. ft

INDUS: Proportion of non-retail business acres per town

CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX: Nitric oxide concentration (parts per 10 million)

RM: Average number of rooms per dwelling

AGE: Proportion of owner-occupied units built prior to 1940

DIS: Weighted distances to five Boston employment centers

RAD: Index of accessibility to radial highways

TAX: Full-value property tax rate per \$10,000

PTRATIO: Pupil-teacher ratio by town

B: $1000(B_k - 0.63)^2$, where B_k is the proportion of [people of African American descent] by town

LSTAT: Percentage of lower status of the population

```
In [2]: # Importing Data
from sklearn.datasets import load_boston
boston_data = load_boston()
```

```
In [3]: boston = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
boston.head()
```

```
Out[3]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
In [4]: #Now we'll add price column
boston['Price'] = boston_data.target
boston.head()
```

```
Out[4]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
In [5]: boston.describe()
```

```
Out[5]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	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	68.574901
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

```
In [6]: boston.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
CRIM      506 non-null float64
ZN        506 non-null float64
INDUS     506 non-null float64
CHAS      506 non-null float64
NOX       506 non-null float64
RM        506 non-null float64
AGE       506 non-null float64
DIS       506 non-null float64
RAD       506 non-null float64
TAX       506 non-null float64
PTRATIO   506 non-null float64
B         506 non-null float64
LSTAT     506 non-null float64
Price     506 non-null float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [7]: *#np.c_: it is used to concatenate columns*

```
X = boston_data.data  
Y = boston_data.target
```

In [8]: `Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, Y, test_size =0.2,
random_state = 0)`

```
print("xtrain shape : ", Xtrain.shape)  
print("xtest shape : ", Xtest.shape)  
print("ytrain shape : ", Ytrain.shape)  
print("ytest shape : ", Ytest.shape)
```

```
xtrain shape : (404, 13)  
xtest shape : (102, 13)  
ytrain shape : (404,)  
ytest shape : (102,)
```

In [9]:

```
Ytrain = Ytrain.astype(int)
```

In [10]:

```
lr = SGDClassifier(loss="hinge", penalty="l2", max_iter=1500)  
lr.fit(Xtrain, Ytrain)  
  
# predicting the test set results  
Y_pred = lr.predict(Xtest)
```

In [11]: *# Results of Linear Regression.*

```
from sklearn import metrics  
from sklearn.metrics import r2_score  
print('Mean Absolute Error : ', metrics.mean_absolute_error(Ytest, Y_pred))  
print('Mean Square Error : ', metrics.mean_squared_error(Ytest, Y_pred))  
print('RMSE', np.sqrt(metrics.mean_squared_error(Ytest, Y_pred)))  
print('R squared error', r2_score(Ytest, Y_pred))
```

```
Mean Absolute Error : 5.641176470588235  
Mean Square Error : 89.84921568627448  
RMSE 9.478882618023839  
R squared error -0.10341321436818918
```

In []: