Machine_Learning_2

March 14, 2019

1 Build the linear regression model using scikit learn in boston data to predict 'Price' based on other dependent variable.

In [1]:
```
# Loading the required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston

boston = load_boston()

# As the boston data is in form of dictionaries in sklearn,let's convert that into
# Data Frame using Pandas.

bos = pd.DataFrame(boston.data)
bos.head()
```

Out[1]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
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 [2]:
```
# Here we don't have the column names.
# The bost data in sklearn has the feature names as another dictionary.

print(boston.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [3]:
```
# now let's add the feature names as columns in our data frame bos.

bos.columns = boston.feature_names
bos.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]:
```
# Here we are missing the price column.It is there as a separate dictionary called tar
# Let's add the price column to bos

bos['Price'] = boston.target
bos.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
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	24.0
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	21.6
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	34.7
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	33.4
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	36.2

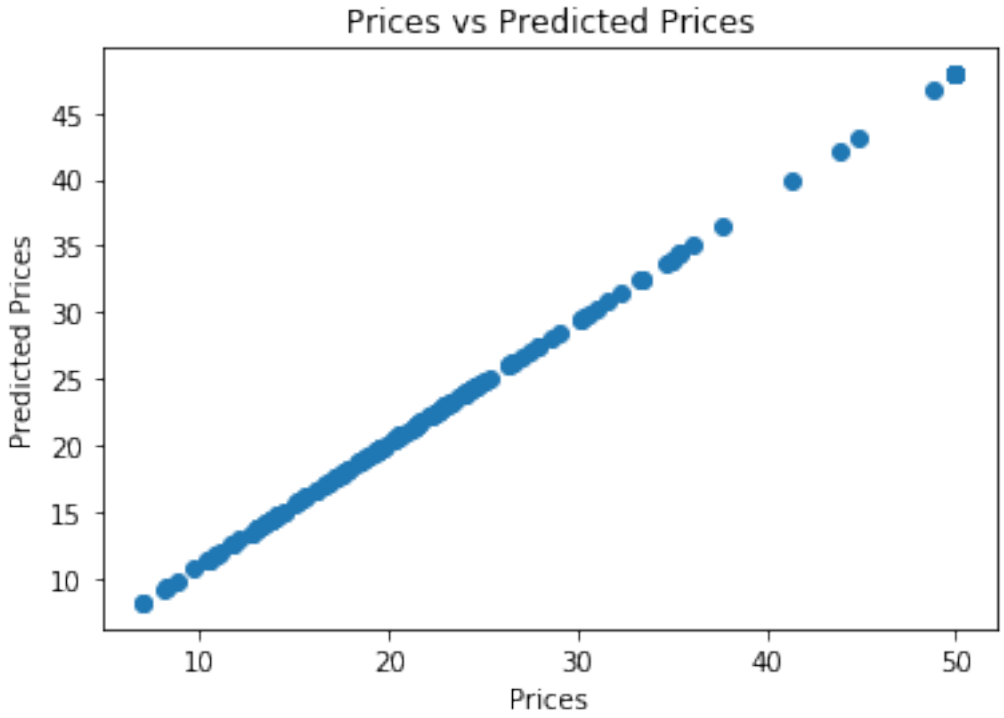
In [5]:
```
# Lets see the data information

bos.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: 55.4 KB
```

In [8]:
```
# Now we can divide our data to dependent and independent variables.

bos['price'] = boston.target
X = bos.drop('price',axis=1).values
y = bos['price'].values

from sklearn.model_selection import train_test_split

# Splitting the data into training and test models

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=5)

# As the data has so many features may be in different units,
# Standard Scaling will give us good results.

from sklearn.preprocessing import StandardScaler
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.fit_transform(X_test)


# Applying the Linear Regressor for our train and test models

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
y_pred=regressor.predict(X_test)

# Let's visualize how our machine is trained and how much is the error
```

```
plt.scatter(y_test,y_pred)
plt.title("Prices vs Predicted Prices")
plt.xlabel("Prices")
plt.ylabel("Predicted Prices")
plt.show()
```



In [9]:
```
# From the above we can observe that the scattered plots falling almost in a straight
# That mean out model is trained with a good accuracy.

#Let's find out the accoracy score.

from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred)
score
```

Out[9]: 0.9940361609655216