Lab Assignment no 4

Aim: Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset. The Boston Housing dataset contains information about various houses in Boston through differen parameters. There are 506 samples and 14 feature variables in this dataset. The objective is to predict the value of prices of the house using the given feature

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

In [2]:
```
x=np.array([95,85,80,70,60])
y=np.array([85,95,70,65,70])
```

In [3]:
```
model= np.polyfit(x, y, 1)
```

In [4]:
```
model
```

Out[4]:
```
array([ 0.64383562, 26.78082192])
```

In [5]:
```
predict = np.poly1d(model)
predict(65)
```

Out[5]:
```
68.63013698630137
```

In [6]:
```
y_pred= predict(x)
y_pred

```

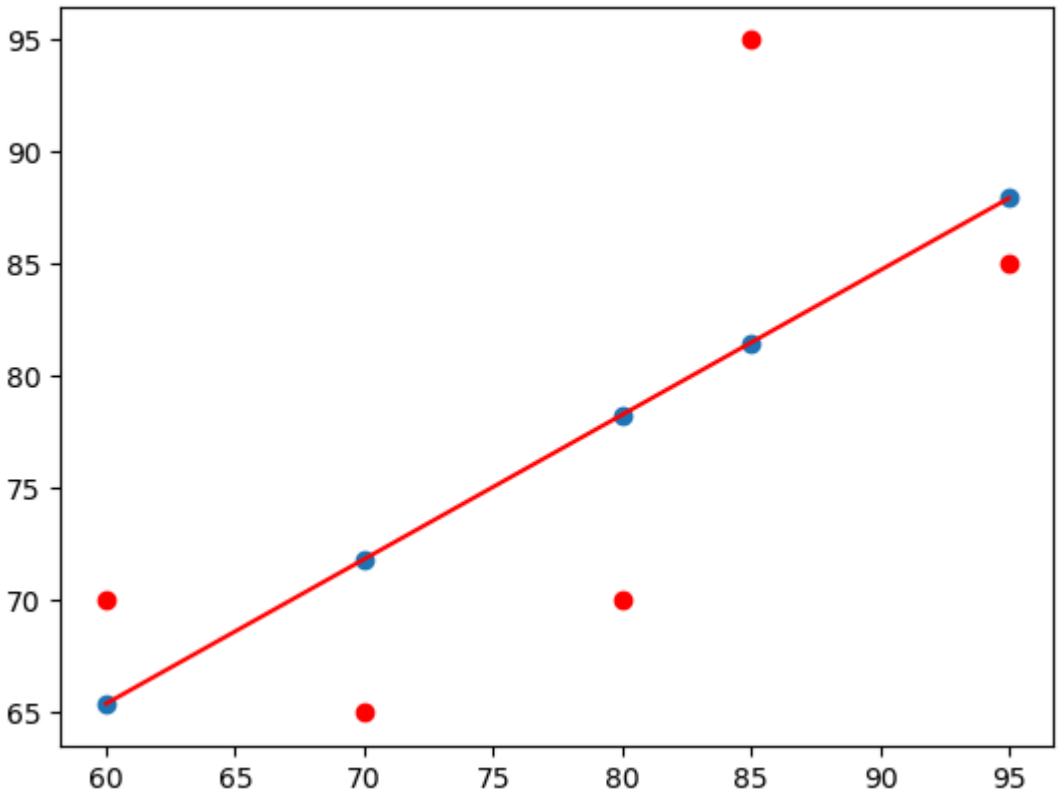
Out[6]:
```
array([87.94520548, 81.50684932, 78.28767123, 71.84931507, 65.4109589 ])
```

In [7]:
```
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

Out[7]:
```
0.4803218090889326
```

In [8]:
```
y_line = model[1] + model[0]* x
plt.plot(x, y_line, c = 'r')
plt.scatter(x, y_pred)
plt.scatter(x,y,c='r')
```

Out[8]: <matplotlib.collections.PathCollection at 0x1e4b6f6fa90>



In [10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
california_housing = fetch_california_housing()
```

In [19]:
```
data = pd.DataFrame(california_housing.data, columns=california_housing
data['PRICE'] = california_housing.target
```

In [20]:
```
print(data.head())

```

```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude
\
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  PRICE
0    -122.23  4.526
1    -122.22  3.585
2    -122.24  3.521
3    -122.25  3.413
4    -122.25  3.422
```

In [21]:
```
1 data
```

Out[21]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitu
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.

20640 rows × 9 columns

In [22]:
```
1 data.isnull().sum()
```

Out[22]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
PRICE         0
dtype: int64
```

In [23]:
```
1 x = data.drop(['PRICE'], axis = 1)
2 y = data['PRICE']
```

In [25]:
```
1 from sklearn.model_selection import train_test_split
2 xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size =0.2,ran
```

In [27]:
```
1 import sklearn
2 from sklearn.linear_model import LinearRegression
3 lm = LinearRegression()
4 model=lm.fit(xtrain,ytrain)
```

In [28]:
```
1 xtrain
```

Out[28]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitu
12069	4.2386	6.0	7.723077	1.169231	228.0	3.507692	33.83	-117.
15925	4.3898	52.0	5.326622	1.100671	1485.0	3.322148	37.73	-122.
11162	3.9333	26.0	4.668478	1.046196	1022.0	2.777174	33.83	-118.
4904	1.4653	38.0	3.383495	1.009709	749.0	3.635922	34.01	-118.
4683	3.1765	52.0	4.119792	1.043403	1135.0	1.970486	34.08	-118.
...	...	...	...	...	...	...	...	
13123	4.4125	20.0	6.000000	1.045662	712.0	3.251142	38.27	-121.
19648	2.9135	27.0	5.349282	0.933014	647.0	3.095694	37.48	-120.
9845	3.1977	31.0	3.641221	0.941476	704.0	1.791349	36.58	-121.
10799	5.6315	34.0	4.540598	1.064103	1052.0	2.247863	33.62	-117.
2732	1.3882	15.0	3.929530	1.100671	1024.0	3.436242	32.80	-115.

16512 rows × 8 columns

In [29]:
```
1 ytrain_pred=lm.predict(xtrain)
2 ytest_pred=lm.predict(xtest)
```

In [30]:
```
1 testdata=[[0.00632,18.0,2.31,0.0,0.538,6.575,65.2,4.0900,1.0,296.0,15.3
```

In [32]:
```
1 df1=pd.DataFrame(ytrain_pred,ytrain)
2 df2=pd.DataFrame(ytest_pred,ytest)
3 df1
```

Out[32]:

	0
PRICE	
5.00001	1.725911
2.70000	2.885439
1.96100	2.200646
1.18800	1.382820
2.25000	2.220702
...	...
1.44600	1.765119
1.59400	1.351502
2.89300	2.508907
4.84600	3.094513
0.69400	0.472337

16512 rows × 1 columns

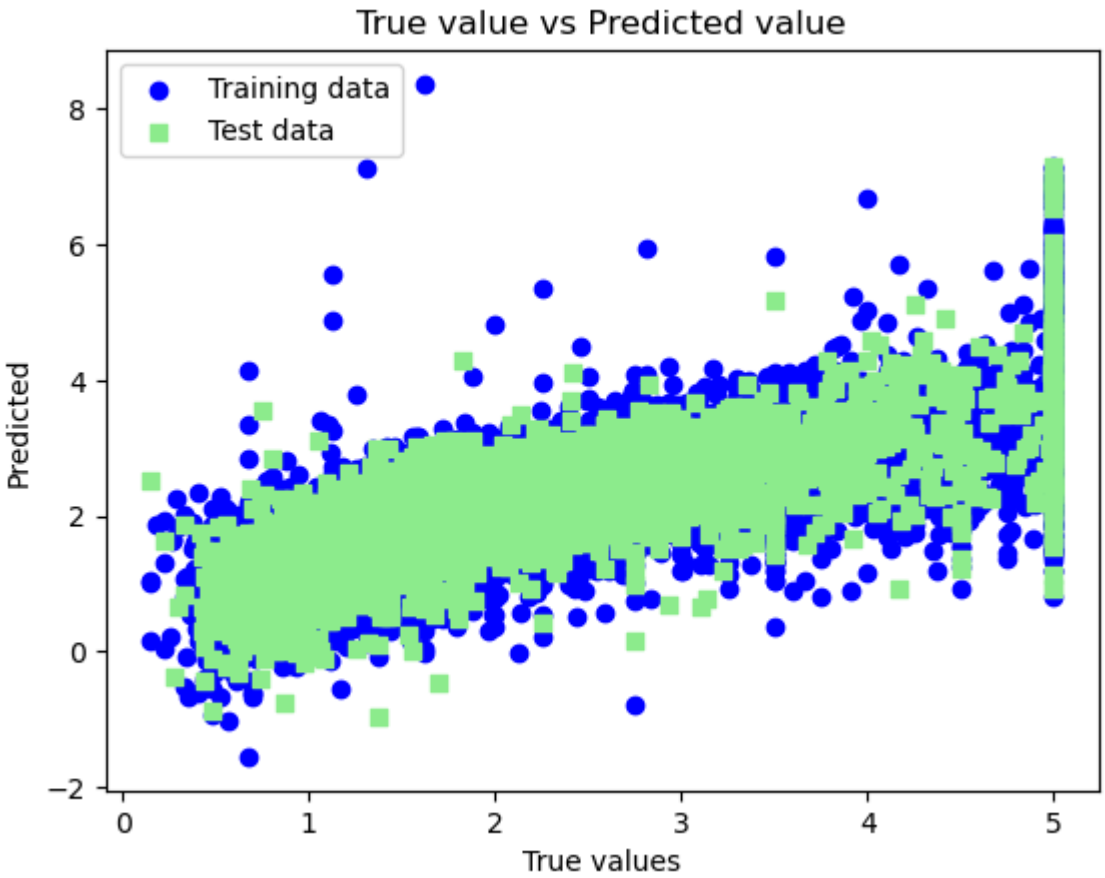
In [33]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print('MSE on test data:',mse)
mse1 = mean_squared_error(ytrain_pred, ytrain)
print('MSE on training data:',mse1)
```

```
MSE on test data: 0.5289841670367209
MSE on training data: 0.5234413607125448
```

In [34]:
```
r2 = lm.score(xtest, ytest)
rmse = (np.sqrt(mean_squared_error(ytest, ytest_pred)))
print('r-squared: {}' .format(r2))
print('----------------------------------------')
print('root mean squared error: {}'.format(rmse))
```

```
r-squared: 0.5943232652466202
----------------------------------------
root mean squared error: 0.7273129773603114
```

In [35]:
```
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training dat
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test dat
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left') #plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



Name:Devesh Kashikar
Roll.no:13217

In [ ]:
```
1
```