Assignment no 4

In [ ]:
```
AIM:To learn about
    1. Linear Regression : Univariate and Multivariate
    2. Least Square Method for Linear Regression
    3. Measuring Performance of Linear Regression
    4. Example of Linear Regression
    5. Training data set and Testing data set:
```

In [68]:
```
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 [4]:
```
model= np.polyfit(x, y, 1)
model
```

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

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

Out[5]:
```
68.63013698630135
```

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

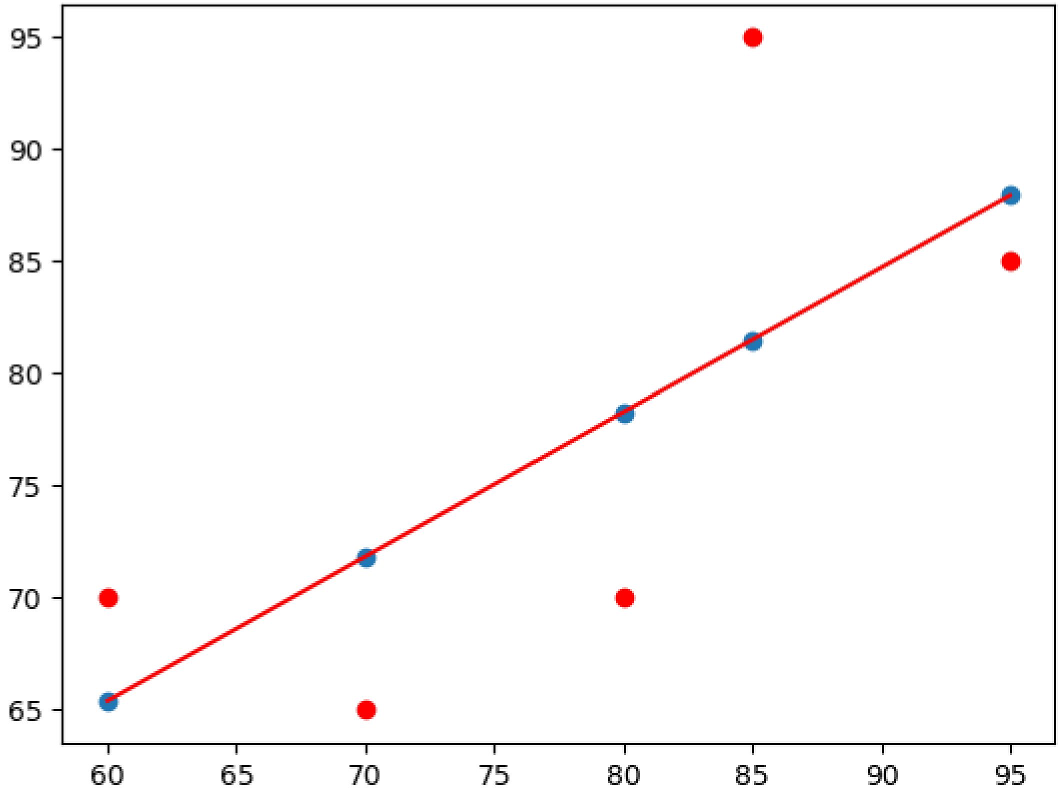
Out[6]:
```
array([87.94520548, 81.50684932, 78.28767123, 71.84931507, 65.4109589 ])
```

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

Out[7]:
```
0.4803218090889323
```

In [16]:
```
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[16]:
```
<matplotlib.collections.PathCollection at 0x27ac8e811f0>
```



In [13]:
```
from sklearn.datasets import fetch_openml
housing = fetch_openml(name="house_prices", as_frame=True)
```

In [14]:
```
data=pd.DataFrame(housing.data)
```

In [15]:
```
data.columns = housing.feature_names
data.head()
```

Out[15]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandCon
0	1	60	RL	65.0	8450	Pave	NaN	Reg	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	

5 rows × 80 columns

In [1]:
```
from sklearn.datasets import fetch_openml
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
housing
```

Out[1]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.5555555
6,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n-----
---------------------\n\n**Data Set Characteristics:**\n\n:Number of Instances:
20640\n\n:Number of Attributes: 8 numeric, predictive attributes and the target
\n\n:Attribute Information:\n    - MedInc        median income in block group\n
- HouseAge      median house age in block group\n    - AveRooms      average nu mber
of rooms per household\n  - AveBedrms   average number of bedrooms per household\n
- Population    block group population\n     - AveOccup       aver age number of
household members\n     - Latitude       block group latitude\n
- Longitude      block group longitude\n\n:Missing Attribute Values: None\n\nThi
s dataset was obtained from the StatLib repository.\nhttps://www.dcc.fc.up.pt/~
ltorgo/Regression/cal_housing.html\n\nThe target variable is the median house v
alue for California districts,\nexpressed in hundreds of thousands of dollars
($100,000).\n\nThis dataset was derived from the 1990 U.S. census, using one ro
w per census\nblock group. A block group is the smallest geographical unit for
which the U.S.\nCensus Bureau publishes sample data (a block group typically ha
s a population\nof 600 to 3,000 people).\n\nA household is a group of people re
siding within a home. Since the average\nnumber of rooms and bedrooms in this d
ataset are provided per household, these\ncolumns may take surprisingly large v
alues for block groups with few households\nand many empty houses, such as vaca
tion resorts.\n\nIt can be downloaded/loaded using the\n:func:`sklearn.dataset
s.fetch_california_housing` function.\n\n.. rubric:: References\n\n- Pace, R. K
elley and Ronald Barry, Sparse Spatial Autoregressions,\n  Statistics and Proba
bility Letters, 33 (1997) 291-297\n'}
```

In [13]:
```python
import pandas as pd
df=pd.DataFrame(housing.data,columns=housing.feature_names)
df
```

Out[13]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
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	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

20640 rows × 8 columns

In [15]:

```
df.head()
```

Out[15]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longit
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-12
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-12
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-12
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-12
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-12

In [19]:

```
df['PRICE'] = housing.target
df
```

Out[19]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
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	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

20640 rows × 9 columns

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
PRICE         0
dtype: int64
```

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

In [25]:
```
x
```

Out[25]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
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	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

20640 rows × 8 columns

In [27]:

```
y
```

Out[27]:

```
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: PRICE, Length: 20640, dtype: float64
```

In [31]:

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_stat
```

In [33]:

```
xtrain
```

Out[33]:

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

16512 rows × 8 columns

In [35]:
```
xtest
```

Out[35]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
14740	4.1518	22.0	5.663073	1.075472	1551.0	4.180593	32.58	
10101	5.7796	32.0	6.107226	0.927739	1296.0	3.020979	33.92	
20566	4.3487	29.0	5.930712	1.026217	1554.0	2.910112	38.65	
2670	2.4511	37.0	4.992958	1.316901	390.0	2.746479	33.20	
15709	5.0049	25.0	4.319261	1.039578	649.0	1.712401	37.79	
...	...	...	...	...	...	...	...	
6655	2.4817	33.0	3.875723	1.034682	2050.0	2.962428	34.16	
3505	4.3839	36.0	5.283636	0.981818	808.0	2.938182	34.25	
1919	3.2027	11.0	5.276074	1.058282	850.0	2.607362	38.86	
1450	6.1436	18.0	7.323529	1.050802	1072.0	2.866310	37.96	
4148	3.3326	52.0	3.891626	1.049261	1462.0	3.600985	34.12	

4128 rows × 8 columns

In [37]:
```
ytrain
```

Out[37]:
```
12069    5.00001
15925    2.70000
11162    1.96100
4904     1.18800
4683     2.25000
          ...
13123    1.44600
19648    1.59400
9845     2.89300
10799    4.84600
2732     0.69400
Name: PRICE, Length: 16512, dtype: float64
```

In [39]:
```
ytest
```

Out[39]:
```
14740    1.369
10101    2.413
20566    2.007
2670     0.725
15709    4.600
         ...
6655     1.695
3505     2.046
1919     1.286
1450     2.595
4148     1.676
Name: PRICE, Length: 4128, dtype: float64
```

In [41]:
```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

In [50]:
```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

In [52]:
```
ytrain_pred
```

Out[52]:
```
array([1.7259112 , 2.88543882, 2.20064594, ..., 2.50890725, 3.0945134 ,
       0.47233661])
```

In [54]:
```
ytest_pred
```

Out[54]:
```
array([2.28110738, 2.79009128, 1.90332794, ..., 0.8418697 , 2.7984953 ,
       2.21779325])
```

In [56]:
```
df=pd.DataFrame(ytrain_pred,ytrain)
df
```

Out[56]:

	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 [58]:
```
df=pd.DataFrame(ytest_pred,ytest)
df
```

Out[58]:

	0
PRICE	
1.369	2.281107
2.413	2.790091
2.007	1.903328
0.725	1.017603
4.600	2.948524
...	...
1.695	1.616753
2.046	2.409188
1.286	0.841870
2.595	2.798495
1.676	2.217793

4128 rows × 1 columns

In [60]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

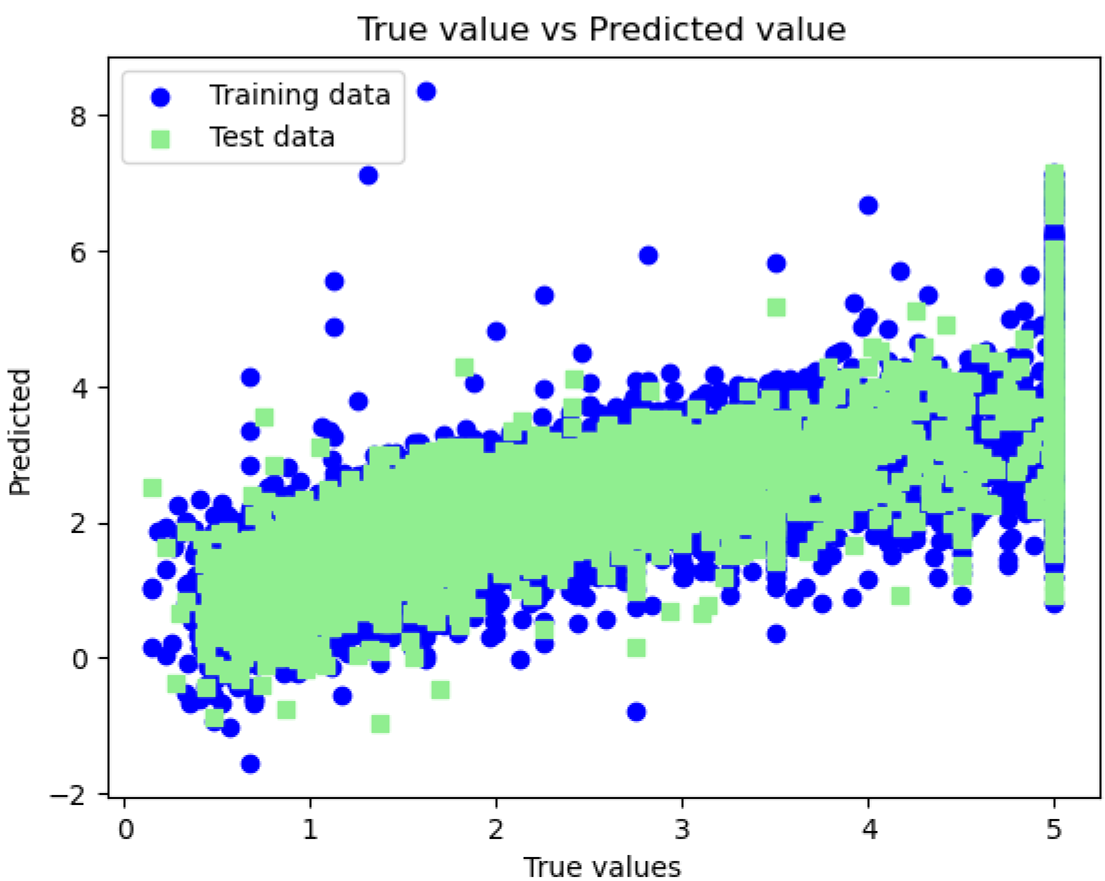
```
mse = mean_squared_error(ytrain_pred,ytrain)
print(mse)
```

```
0.5289841670367224
0.5234413607125449
```

In
```
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

```
0.5289841670367224
```

In
```
plt.scatter(ytrain  ,ytrain_pred,c='blue',marker='o',label='Training  data')
plt.scatter(ytest,ytest_pred  ,c='lightgreen',marker='s',label='Test  data')
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 : Sohan Mardhekar
Roll No : 13230
Batch : B2