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
```

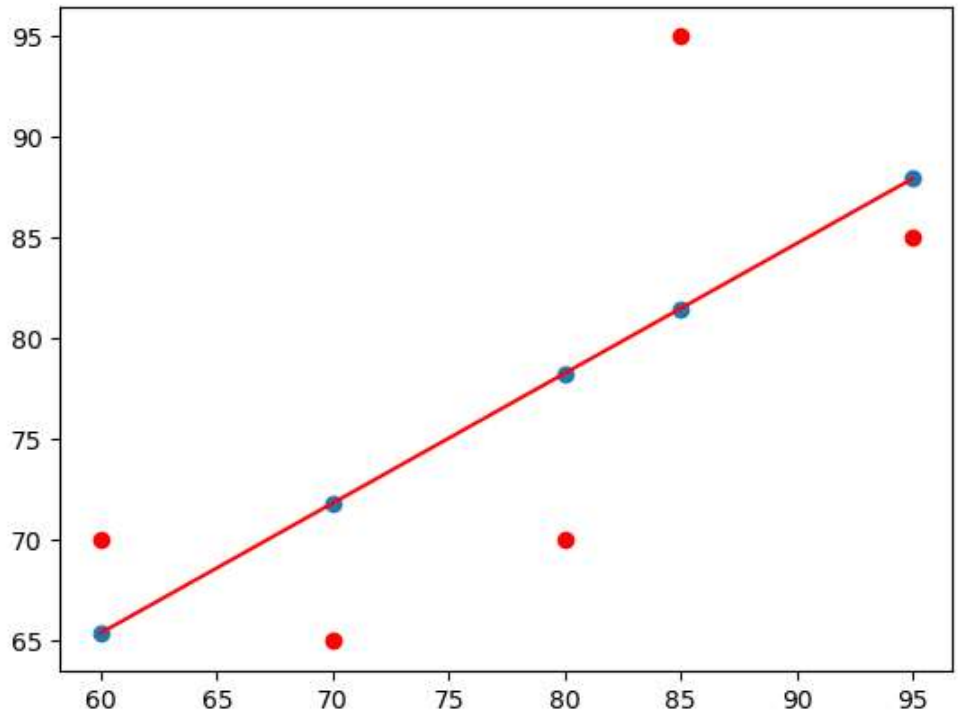
Out[6]:
```
array([87.94520548, 81.50684932, 78.28767123, 71.84931507, 65.4109589 ])
```

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 0x261e8675850>
```



In [9]:
```
from sklearn.datasets import fetch_openml
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

In [10]:
```
housing
```

Out[10]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.55555556,
           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    :Num
ber of Attributes: 8 numeric, predictive attributes and the target\n\n    :Attribute Infor
mation:\n        - MedInc        median income in block group\n        - HouseAge      med
ian house age in block group\n        - AveRooms      average number of rooms per househol
d\n        - AveBedrms     average number of bedrooms per household\n        - Population
block group population\n        - AveOccup      average number of household members\n
- Latitude      block group latitude\n        - Longitude     block group longitude\n\n
:Missing Attribute Values: None\n\nThis dataset was obtained from the StatLib repositor
y.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe target variable is
the median house value for California districts,\nexpressed in hundreds of thousands of do
llars ($100,000).\n\nThis dataset was derived from the 1990 U.S. census, using one row per
census\nblock group. A block group is the smallest geographical unit for which the U.S.\nC
ensus Bureau publishes sample data (a block group typically has a population\nof 600 to 3,
000 people).\n\nA household is a group of people residing within a home. Since the average
\nnumber of rooms and bedrooms in this dataset are provided per household, these\ncolumns
may take surprisingly large values for block groups with few households\nand many empty ho
uses, such as vacation resorts.\n\nIt can be downloaded/loaded using the\n:func:`sklearn.d
atasets.fetch_california_housing` function.\n\n.. topic:: References\n\n    - Pace, R. Kel
ley and Ronald Barry, Sparse Spatial Autoregressions,\n      Statistics and Probability Le
tters, 33 (1997) 291-297\n'}
```

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

Out[13]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

In [15]:
```
target=housing.target_names
```

In [16]:
```
target
```

Out[16]:
```
['MedHouseVal']
```

In [18]:
```
data1 = pd.DataFrame(data=np.c_[housing ['data'], housing ['target']],
                                columns= housing ['feature_names'] + ['target'])

```

In [23]:
```
from sklearn.datasets import fetch_openml
housing = fetch_openml(name = 'house_prices', as_frame = True)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\datasets\_openml.py:1002: FutureWarnin
g: The default value of `parser` will change from `'liac-arff'` to `'auto'` in 1.4. You ca
n set `parser='auto'` to silence this warning. Therefore, an `ImportError` will be raised
from 1.4 if the dataset is dense and pandas is not installed. Note that the pandas parser
may return different data types. See the Notes Section in fetch_openml's API doc for detai
ls.
  warn(
```

In [24]:
```
df
```

Out[24]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

In [25]:
```
data1
```

Out[25]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...	...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

In [26]:
```
df.isnull().sum()
```

Out[26]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
dtype: int64
```

In [27]:
```
x = data1.drop(['target'], axis = 1)
y = data1['target']
```

In [29]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 0
```

In [30]:
```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

In [31]:
```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

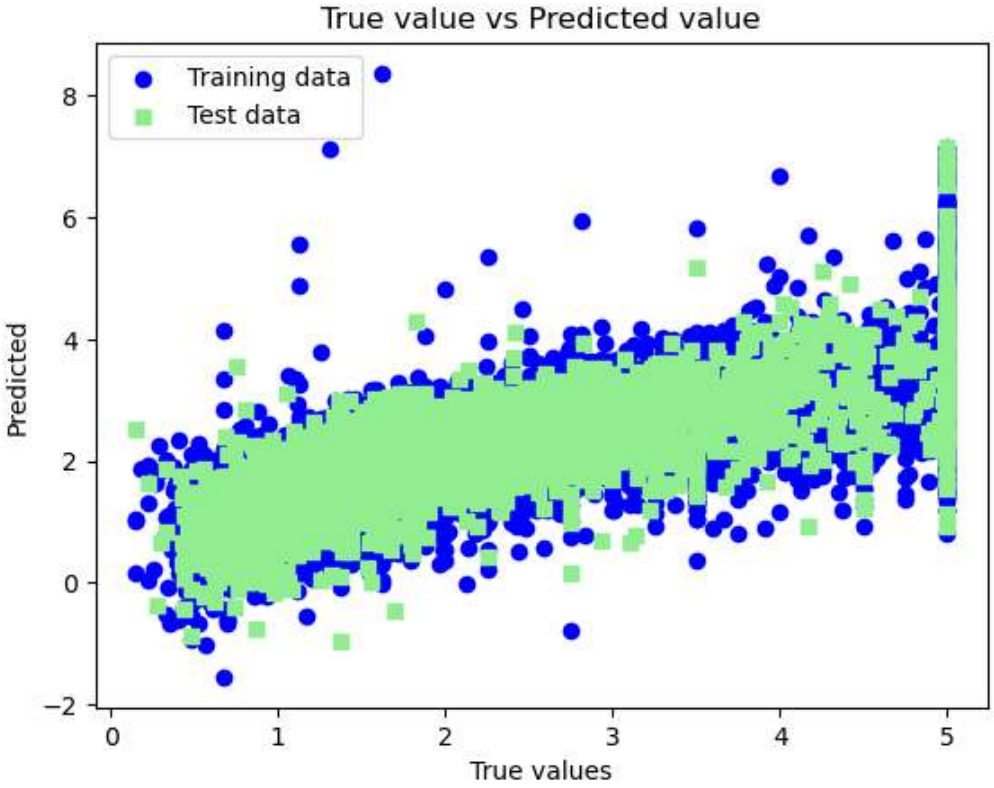
In [32]:
```
df=pd.DataFrame(ytrain_pred,ytrain)
df=pd.DataFrame(ytest_pred,ytest)
```

In [33]:
```
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)
```

```
0.5289841670367192
0.5234413607125448
```

In [34]:
```
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()

```




```
# Swayambhu Bhapkar
# Roll no:-13121
```