DSBDA Lab Assignment No. 4

```
Name: Akash Ganesh Padir
Roll No.: TEB04
```

In [61]:
```
#Step 1:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [62]:
```
#Step 2:
x=np.array([1,2,3,4,5])
y=np.array([3,4,2,4,5])
```

In [63]:
```
#Step 3:
model=np.polyfit(x,y,1)
```

In [64]:
```
#Step 43:
model
```

Out[64]:
```
array([0.4, 2.4])
```

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

Out[65]:
```
4.4
```

In [66]:
```
predict= np.poly1d(model)
predict(4)
```

Out[66]:
```
4.0
```

In [67]:
```
#Step 6:
y_pred=predict(x)
y_pred
```

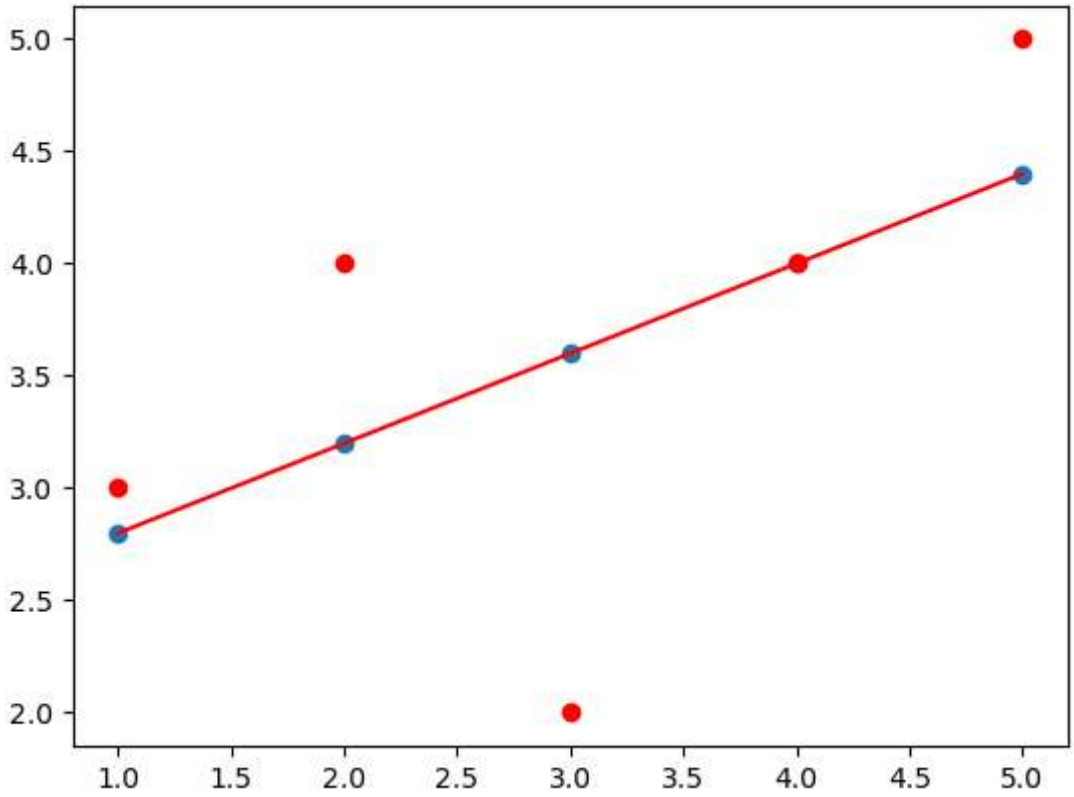
Out[67]:
```
array([2.8, 3.2, 3.6, 4. , 4.4])
```

In [68]:
```
#Step 7:
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

Out[68]:
```
0.30769230769230793
```

In [69]:
```
#Step 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[69]: <matplotlib.collections.PathCollection at 0x200b07508b0>

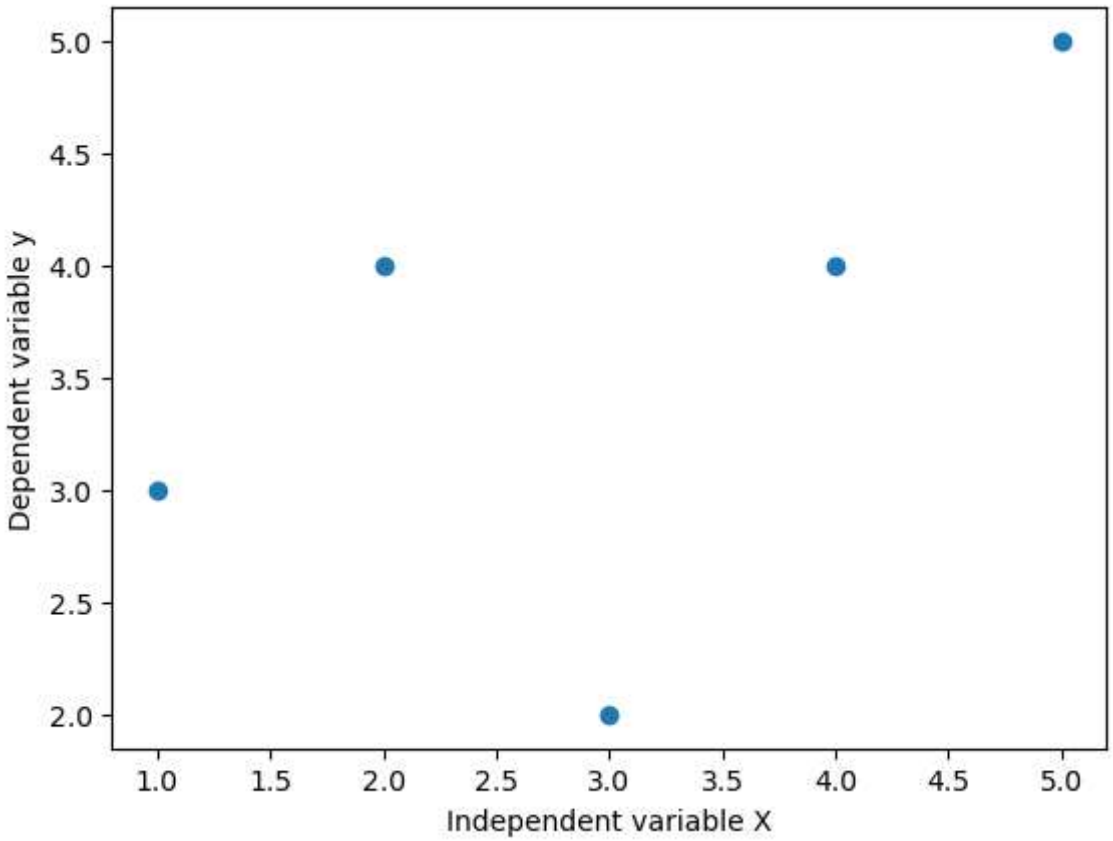


In [70]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
```

In [71]:
```
x = np.array([1,2,3,4,5])
y = np.array([3,4,2,4,5])
#y = np.array([7,14,15,18,19])
n = np.size(x)

x_mean = np.mean(x)
y_mean = np.mean(y)
x_mean,y_mean

Sxy = np.sum(x*y)- n*x_mean*y_mean
Sxx = np.sum(x*x)-n*x_mean*x_mean

b1 = Sxy/Sxx
b0 = y_mean-b1*x_mean
print('slope b1 is', b1)
print('intercept b0 is', b0)

plt.scatter(x,y)
plt.xlabel('Independent variable X')
plt.ylabel('Dependent variable y')
```

```
slope b1 is 0.4
intercept b0 is 2.4
```

Out[71]:
```
Text(0, 0.5, 'Dependent variable y')
```



In [72]:
```
error = y - y_pred
se = np.sum(error**2)
print('squared error is', se)

mse = se/n
print('mean squared error is', mse)

rmse = np.sqrt(mse)
print('root mean square error is', rmse)

SSt = np.sum((y - y_mean)**2)
R2 = 1- (se/SSt)
print('R square is', R2)
```

```
squared error is 3.5999999999999988
mean squared error is 0.7199999999999998
root mean square error is 0.8485281374238569
R square is 0.30769230769230793
```

In [73]:
```
x = x.reshape(-1,1)
regression_model = LinearRegression()
# Fit the data(train the model)
regression_model.fit(x, y)
# Predict
y_predicted = regression_model.predict(x)
# model evaluation
mse=mean_squared_error(y,y_predicted)
rmse = np.sqrt(mean_squared_error(y, y_predicted))
r2 = r2_score(y, y_predicted)
# printing values
print('Slope:' ,regression_model.coef_)
print('Intercept:', regression_model.intercept_)
print('MSE:',mse)
print('Root mean squared error: ', rmse)
print('R2 score: ', r2)
```

```
Slope: [0.4]
Intercept: 2.4
MSE: 0.72
Root mean squared error:  0.848528137423857
R2 score:  0.3076923076923078
```

In [74]:
```
#(Boston Dataset):
#Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [75]:
```
from sklearn.datasets import load_boston
boston = load_boston()
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: F
utureWarning: Function load_boston is deprecated; `load_boston` is deprecated
in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer t
o
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of thi
s
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

In [76]:
```
#step 3:
data = pd.DataFrame(boston.data)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       506 non-null    float64
 1   1       506 non-null    float64
 2   2       506 non-null    float64
 3   3       506 non-null    float64
 4   4       506 non-null    float64
 5   5       506 non-null    float64
 6   6       506 non-null    float64
 7   7       506 non-null    float64
 8   8       506 non-null    float64
 9   9       506 non-null    float64
 10  10      506 non-null    float64
 11  11      506 non-null    float64
 12  12      506 non-null    float64
dtypes: float64(13)
memory usage: 51.5 KB
```

In [77]:
```
#step 4:
data.columns = boston.feature_names
data.head()
```

Out[77]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
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	4.9
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	9.1
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	4.0
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	2.9
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	5.3

In [78]:
```
#step 5:
data['PRICE'] = boston.target
```

In [79]:
```
#step 6:
data.isnull().sum()
```

Out[79]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

In [80]:
```
#step 7:
x = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

In [81]:
```
#step 8:
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_st
```

In [82]:
```
xtrain.shape, ytrain.shape
```

Out[82]:
```
((404, 13), (404,))
```

In [83]:
```
xtest.shape, ytest.shape
```

Out[83]:
```
((102, 13), (102,))
```

In [84]:
```
#step 9:
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

In [85]:
```
#step 10:
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

In [86]:
```
ytrain_pred
```

Out[86]:
```
array([32.55692655, 21.92709478, 27.54382573, 23.60318829,  6.57190962,
       14.94183849, 22.2234359 , 29.16492082, 33.24362083, 13.14592261,
       20.25607099, 20.69823381, 12.65147525, 23.36451045,  5.04647867,
       19.82921197,  9.41949932, 44.64390988, 30.78308135, 12.51377155,
       17.7083025 , 21.40137495, 23.63206936, 20.43451195, 35.01471208,
       13.84093827, 21.04977584, 35.15299117, 19.43031106, 13.17488144,
       14.10200042, 23.10677783, 14.38600111, 31.24428679, 25.30231549,
       15.41257398, 24.21291852,  9.40801187, 14.94526286, 20.83029825,
       32.74172958, 27.96372521, 25.60836003, 15.56419667, 31.11934684,
       27.96958264, 13.99703059,  7.63346533, 28.4388332 , 25.33766463,
        4.52504654, 28.38514306, 17.1896917 , 29.74225124, 20.45365104,
       15.92613078, 17.88247152, 12.73233004,  8.75151422, 19.2087374 ,
       34.49694507, 32.94684483, 23.67278817, 19.55243904, 22.8357545 ,
       26.87133257, 21.80817968, 17.06379885, 32.05027982, 10.92397211,
       19.43423447, 32.4854791 , 18.83330461, 15.95730389, 18.64348601,
       14.44808929, 24.60654801, 24.2966726 , 16.64095381, 13.32850391,
       20.20307548, 25.12819701, 17.18033172, 24.71277155, 22.55275499,
       27.95373582, 35.65590799, 16.64554264, 11.83311357, 34.84466464,
       30.84970933, 20.7296176 , 39.5623948 , 28.93322544, 29.14486603,
       17.37121002, 26.82268232, 40.00777296, 28.73960914, 16.44453732,
       37.45185446, 35.50108073, 13.44578945, 29.15098204, 21.60750842,
       24.3179916 , 21.41700241, 23.69538029, 27.763419  , 29.66227826,
       14.17302558, 26.07579718, 23.29927812, 12.80163317, 13.72880538,
       25.27684715, 19.3372779 , 30.54665354, 10.97447089, 23.60361618,
       16.97107603, 16.94075184, 22.59508311, 21.66478168, 11.77477027,
       25.21624705, 28.69690945, 20.17018883, 12.57893016, 25.48767672,
       25.94576428, 25.07919075, 23.5616099 , 26.7499689 , 16.61402974,
       21.79867747, 36.15143711, 21.00423145, 35.88524905, 25.69352037,
       21.5263148 , 15.87068763, 31.29616772, 21.21153127, 27.77524582,
       14.8263031 , 32.22158358, 13.99145209,  1.72558788, 19.37012454,
       14.26927105, 37.54465846, 15.72768892, 14.42603011, 27.31195528,
       23.24522425, 18.47439387, 30.56792527, 27.27498194, 27.27933163,
       24.82223745, 24.16626145, 23.72500963, 11.15226922, 20.76322385,
       13.54743953, 17.16753222, 12.72059151, 28.36113417, 14.93078086,
       16.28718393, 28.70785889, 14.89693976, 21.24395164, 12.83793534,
       13.8967354 , 22.63435472, 21.22168525, 14.71193886, 20.93690941,
       16.89161444, 24.57078637, 12.55171427, 34.77581569, 12.04428697,
       43.13783582, 31.24743877, 35.27489214, 21.44652404, 15.75342369,
       26.54541539, 29.48749252, 14.09267258, 26.55382087, 37.06264306,
       17.64994791, 10.60033751, 34.12962592, 35.60893841, 18.29850589,
       22.55033558, 17.99336763, 24.37931178, 19.51737003, 27.30545421,
       -4.3921497 , 20.5694959 , 35.24711794, 36.62936652, 25.08667454,
       27.21318383, 20.76260072, 20.62207277, 15.87527321, 20.67111164,
       20.55222254, 27.90614562, 19.6623801 ,  7.40663904, 16.40149348,
       32.41751592, 35.22532239, 17.48615135, 18.73060335, 23.40379308,
        6.90428516, 21.44745461, 24.02200142, 16.46784691, 18.38505179,
       21.90096579, 27.59158204, 25.48139462, 37.02340322, 15.43332269,
       28.60694794, 25.833241  , 22.27537004, 38.70334609, 20.83802332,
       23.4287209 , 22.86380935, 12.48755328, 20.30380995, 33.59657861,
       24.79674983, 18.00283472, 33.54517371, 21.63038303, 28.34884771,
       32.26697938, 36.74735276, 22.21068249, 24.03052252, 22.44265374,
       31.82414277, 22.3672764 , 18.83724841, 21.79697632, 28.24902955,
       22.5282343 , 21.81339391, 17.00781251, 17.49258071, 16.96573172,
       17.42535476, 16.49296072, 31.60388241, 23.76669997, 17.5783377 ,
       19.8104465 , 33.69341038, 13.95441929, 24.95294806, 17.37139503,
       30.49949836, 29.96325775, 22.55730163, 20.82912579, 35.02490097,
       22.62414952, 32.89864678, 20.77381521, 31.41949305, 30.90222525,
```

```
37.56313681, 26.83815938, 21.9299641 , 28.71684915, 16.17264967,
26.97631217, 21.09345616, 30.46198221,  9.94954653, 30.89000499,
 5.84660346, 15.62690795, 18.15511465, 35.40907542, 32.07204745,
11.0533533 , 13.29217059, 21.60325564, 34.44368387, 18.63979788,
19.19398001, 15.00401901, 25.78879807, 41.15008314, 25.03321118,
42.02049754, 24.93655332, 22.29015819, 12.26449688, 12.01986598,
14.14864319, 18.48192539,  3.06216934, 27.51260448, 26.07255247,
41.04860013, 21.10381709, 21.14679988, 34.06176587, 33.41315924,
 9.7266196 , 24.74423086, 43.37659562, 16.9546337 , 17.89698454,
25.51231449, 18.43599095,  6.12378352, 19.32867486, 34.9210476 ,
16.23395668, 23.02767993, 13.57396353, 24.50837677, 18.77408796,
17.32594697, 18.77680161, 33.1153209 , 19.46400572, 30.73370111,
32.76024301, 41.30498546, 19.14302841, 16.57710162, 37.54775334,
17.98685071,  9.44489833, 15.1641066 , 24.94105282, 19.68981249,
16.62354285, 27.42640244, 12.97145628,  5.84069961, 19.01616688,
 9.8593521 , 28.08568859,  4.55202156, 29.19078851, 32.18448197,
22.14626525, 16.77353103, 18.09521326, 20.69880761, 33.59801009,
27.76466052, 19.52622298, 20.73263109,  6.66762852, 28.91184184,
24.61296652, 22.15495216, 13.64649885, 25.7963897 , 19.33474204,
 8.85925152, 26.69406634, 16.19490488, 31.36752127, 32.61895119,
25.44594779, 18.53296899, 30.60523455, 21.56355414, 25.27299928,
25.91044256, 31.59739298, 24.50960565, 34.45005694, 17.11216878,
19.69986884, 18.54092642, 40.99282958, 25.1228036 , 19.49495107,
33.32636427, 23.79620777, 18.45835276, 23.24918114])
```

In [87]:
```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

In [88]:
```
#step 11:
df1=pd.DataFrame(ytrain_pred,ytrain)
df2=pd.DataFrame(ytest_pred,ytest)
```

In [89]:
```
df1
```

Out[89]:

PRICE	0
26.7	32.556927
21.7	21.927095
22.0	27.543826
22.9	23.603188
10.4	6.571910
...	...
18.5	19.494951
36.4	33.326364
19.2	23.796208
16.6	18.458353
23.1	23.249181

404 rows × 1 columns

In [90]:
```
df2
```

Out[90]:

PRICE	0
22.6	24.889638
50.0	23.721411
23.0	29.364999
8.3	12.122386
21.2	21.443823
...	...
24.7	25.442171
14.1	15.571783
18.7	17.937195
28.1	25.305888
19.8	22.373233

102 rows × 1 columns

In [91]:
```
#step 12:
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

```
33.4489799976765
```

In [92]:
```
mse = mean_squared_error(ytrain_pred,ytrain)
print(mse)
```

```
19.326470203585725
```

In [93]:
```
#step 13:
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()
```