ASSIGN_2_PROBLEM1 ¶

Predicting Value of Y given X1 and X2.

1.Enter the values of X1 and X2.

2. Get the predicted value of Y 3.Calculate the value of Root Mean Squared Error

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import statsmodels.api as sm
import numpy as np

df = pd.read_csv("/home/akash/Desktop/github/MACHINE LEARNING/ml_assign2_q1.csv")
X = df[['X1','X2']]
y = df['Y']
regr = linear_model.LinearRegression()
regr.fit(X, y)
x1 = int(input("Enter the value of x1 to predict y\n"))
x2 = int(input("Enter the value of x2 to predict y\n"))
predictedy = regr.predict([[x1,x2]])
print("Predicted value of Y for ",x1,"and ",x2,"is",predictedy)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred
```

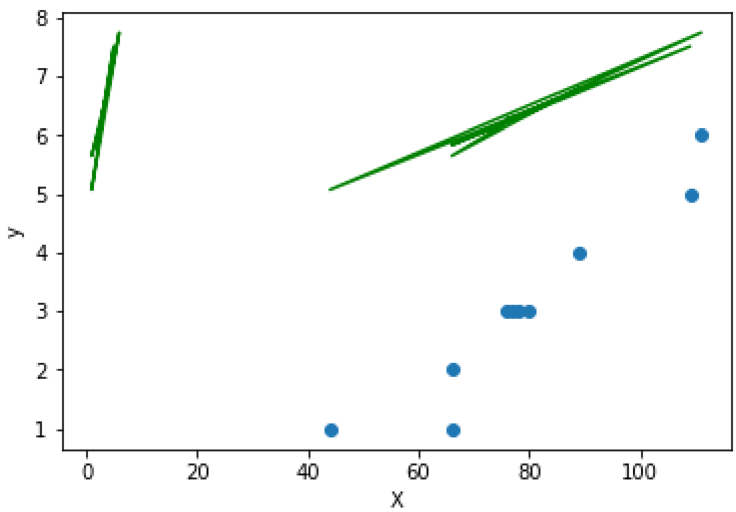
```
Enter the value of x1 to predict y
8
Enter the value of x2 to predict y
5
Predicted value of Y for  8 and  5 is [4.86214125]
Mean Absolute Error: 0.24999999999999956
Mean Squared Error: 0.06499999999999981
Root Mean Squared Error: 0.25495097567963887
```

Calculation of Slope and Intercept Values

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
dataset = pd.read_csv('/home/akash/Desktop/github/MACHINE LEARNING/ml_assign2_q1.cs
dataset.head()
dataset.describe()

from sklearn.linear_model import LinearRegression
X = dataset.drop('Y', axis=1)
y = dataset['Y']
model = LinearRegression()
model.fit(X, y)
model = LinearRegression().fit(X, y)
r_sq = model.score(X, y)
print('coefficient of determination:', r_sq)
print('intercept value of the model is :', model.intercept_)
print('Slope is :', model.coef_)
y_pred = model.predict(X)
print('predicted response:', y_pred, sep='\n')
plt.scatter(X["X1"],X["X2"])
plt.plot(X, y_pred, color = "g")
# putting labels
plt.xlabel('X')
plt.ylabel('y')
# function to show plot
plt.show()
```

```
coefficient of determination: 0.8713995029975861
intercept value of the model is : 3.73215813168261
Slope is : [0.02622257 0.18404052]
predicted response:
[6.80212859 5.64688801 6.32963984 7.74710608 5.06999156 6.30341728
 6.38208497 5.83092853 7.51062043 6.27719471]
```



In [3]:

```
import statsmodels.api as sm
X = sm.add_constant(X) # adding a constant

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
========
Dep. Variable:                      Y   R-squared:
0.871
Model:                            OLS   Adj. R-squared:
0.835
Method:                 Least Squares   F-statistic:
23.72
Date:                Sat, 22 May 2021   Prob (F-statistic):
0.000763
Time:                        22:36:30   Log-Likelihood:
-1.9830
No. Observations:                  10   AIC:
9.966
Df Residuals:                       7   BIC:
10.87
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const          3.7322      0.887      4.208      0.004       1.635
5.830
X1             0.0262      0.020      1.310      0.232      -0.021
0.074
X2             0.1840      0.251      0.733      0.487      -0.409
0.777
==============================================================================
========
Omnibus:                        1.340   Durbin-Watson:
2.402
Prob(Omnibus):                  0.512   Jarque-Bera (JB):
0.867
Skew:                           0.654   Prob(JB):
0.648
Kurtosis:                       2.393   Cond. No.
670.
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
/home/akash/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.
```

```
py:1535: UserWarning: kurtosistest only valid for n>=20 ... continui
ng anyway, n=10
  "anyway, n=%i" % int(n))
```

ASSIGN_2_PROBLEM2

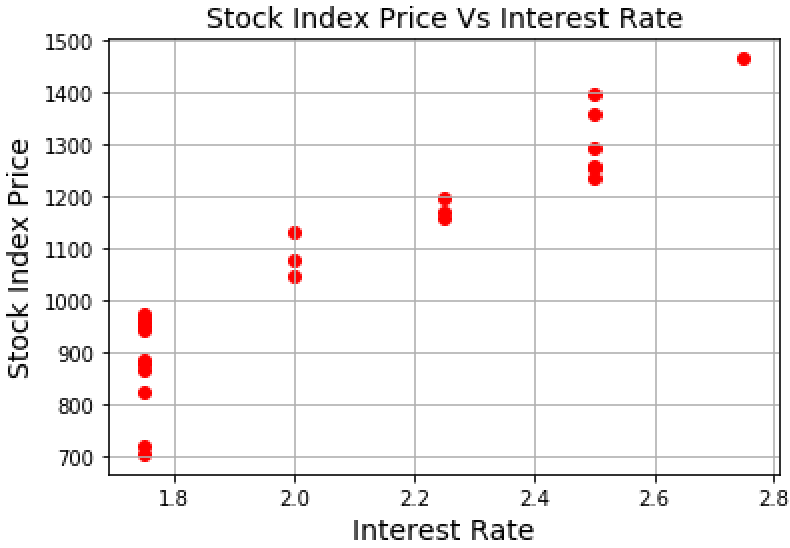
In [4]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import statsmodels.api as sm

Stock_Market = {'Year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,201
                'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
                'Interest_Rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2
                'Unemployment_Rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5
                'Stock_Index_Price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,
                }

df = pd.DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemploymen

print (df)
```

```
    Year  Month  Interest_Rate  Unemployment_Rate  Stock_Index_Price
0   2017     12           2.75                5.3               1464
1   2017     11           2.50                5.3               1394
2   2017     10           2.50                5.3               1357
3   2017      9           2.50                5.3               1293
4   2017      8           2.50                5.4               1256
5   2017      7           2.50                5.6               1254
6   2017      6           2.50                5.5               1234
7   2017      5           2.25                5.5               1195
8   2017      4           2.25                5.5               1159
9   2017      3           2.25                5.6               1167
10  2017      2           2.00                5.7               1130
11  2017      1           2.00                5.9               1075
12  2016     12           2.00                6.0               1047
13  2016     11           1.75                5.9                965
14  2016     10           1.75                5.8                943
15  2016      9           1.75                6.1                958
16  2016      8           1.75                6.2                971
17  2016      7           1.75                6.1                949
18  2016      6           1.75                6.1                884
19  2016      5           1.75                6.1                866
20  2016      4           1.75                5.9                876
21  2016      3           1.75                6.2                822
22  2016      2           1.75                6.2                704
23  2016      1           1.75                6.1                719
```

In [5]:

```
plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'], color='red')
plt.title('Stock Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```



In [6]:

```
plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'], color='green')
plt.title('Stock Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

In [7]:

```
X = df[['Interest_Rate','Unemployment_Rate']] # here we have 2 variables for multip
Y = df['Stock_Index_Price']

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
New_Interest_Rate = 2.75
New_Unemployment_Rate = 5.3
print ('Predicted Stock Index Price: \n', regr.predict([[New_Interest_Rate ,New_Une
```

```
Intercept: 
 1798.403977625855
Coefficients: 
 [ 345.54008701 -250.14657137]
Predicted Stock Index Price: 
 [1422.86238865]
```

In [8]:

```
# with statsmodels
X = sm.add_constant(X) # adding a constant

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
======================================================================
========
Dep. Variable:      Stock_Index_Price   R-squared:
0.898
Model:                            OLS   Adj. R-squared:
0.888
Method:                 Least Squares   F-statistic:
92.07
Date:                Sat, 22 May 2021   Prob (F-statistic):
4.04e-11
Time:                        22:36:33   Log-Likelihood:
-134.61
No. Observations:                  24   AIC:
275.2
Df Residuals:                      21   BIC:
278.8
Df Model:                           2
Covariance Type:            nonrobust
======================================================================
===============
                        coef    std err          t      P>|t|      [0.
025      0.975]
----------------------------------------------------------------------
---------------
const              1798.4040    899.248      2.000      0.059     -71.
685    3668.493
Interest_Rate       345.5401    111.367      3.103      0.005     113.
940     577.140
Unemployment_Rate  -250.1466    117.950     -2.121      0.046    -495.
437      -4.856
======================================================================
========
Omnibus:                        2.691   Durbin-Watson:
0.530
Prob(Omnibus):                  0.260   Jarque-Bera (JB):
1.551
Skew:                          -0.612   Prob(JB):
0.461
Kurtosis:                       3.226   Cond. No.
394.
======================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

ASSIGNMENT2 PROBLEM 3

METHOD 1

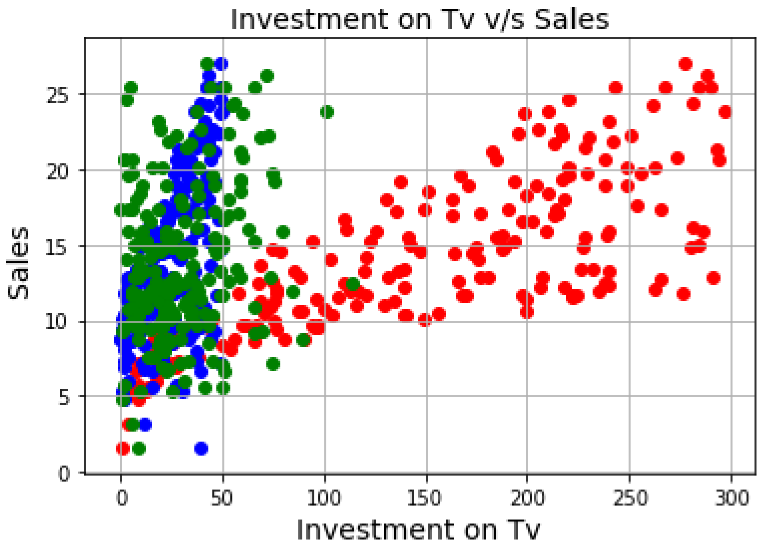
In [9]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import statsmodels.api as sm

data = pd.read_csv("/home/akash/Desktop/github/MACHINE LEARNING/data.csv")
df = pd.DataFrame(data,columns=['TV','radio','newspaper','sales'])
print(df)
```

```
        TV  radio  newspaper  sales
0    230.1   37.8       69.2   22.1
1     44.5   39.3       45.1   10.4
2     17.2   45.9       69.3    9.3
3    151.5   41.3       58.5   18.5
4    180.8   10.8       58.4   12.9
..     ...    ...        ...    ...
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1    9.7
197  177.0    9.3        6.4   12.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   13.4

[200 rows x 4 columns]
```

In [10]:

```
plt.scatter(df['TV'], df['sales'], color='red')
plt.scatter(df['radio'], df['sales'], color='blue')
plt.scatter(df['newspaper'], df['sales'], color='green')
plt.title('Investment on Tv v/s Sales', fontsize=14)
plt.xlabel('Investment on Tv', fontsize=14)
plt.ylabel('Sales ', fontsize=14)
plt.grid(True)
plt.show()
```



In [11]:

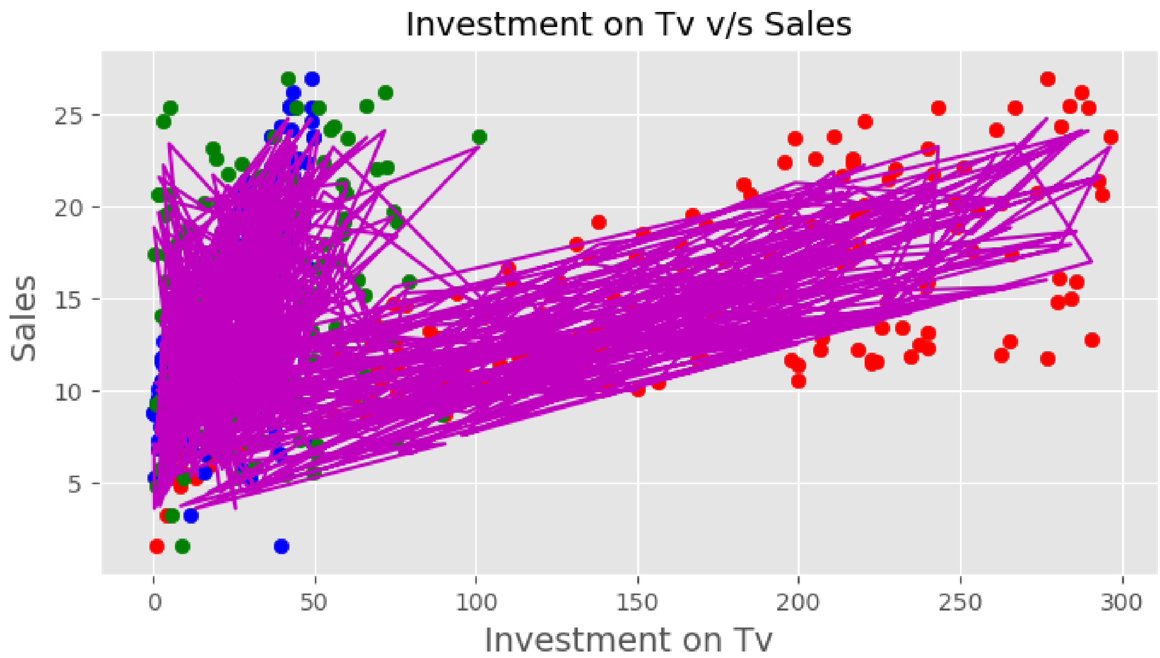
```
X = df[['TV','radio','newspaper']] # here we have 2 variables for multiple regressi
Y = df['sales']

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
TV_invest = float(input("Enter the investment in TV\n"))
radio_invest = float(input("Enter the investment in radio\n"))
newspaper_invest = float(input("Enter the investment in newspaper\n"))
New_Unemployment_Rate = 5.3
print ('Predicted Sales: \n', regr.predict([[TV_invest,radio_invest,newspaper_inves
```

```
Intercept: 
 2.9388893694594085
Coefficients: 
 [ 0.04576465  0.18853002 -0.00103749]
Enter the investment in TV
240
Enter the investment in radio
70
Enter the investment in newspaper
40
Predicted Sales: 
 [27.07800574]
```

REGRESSION LINE

In [24]:

```
ols = linear_model.LinearRegression()
model = ols.fit(X, Y)
response = model.predict(X)
r2 = model.score(X, Y)
plt.style.use('default')
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(8, 4))

#ax.plot(X, response, color='k', label='Regression model')
plt.scatter(df['TV'], df['sales'], color='red')
plt.scatter(df['radio'], df['sales'], color='blue')
plt.plot(X,response,color='m')
plt.scatter(df['newspaper'], df['sales'], color='green')
plt.title('Investment on Tv v/s Sales', fontsize=14)
plt.xlabel('Investment on Tv', fontsize=14)
plt.ylabel('Sales ', fontsize=14)
plt.grid(True)
plt.show()
```



Type *Markdown* and LaTeX: α^2

In [25]:

```
X = df[['TV','radio','newspaper']] # here we have 2 variables for multiple regressi
Y = df['sales']

# with sklearn
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# prediction with sklearn
TV_invest = float(input("Enter the investment in TV\n"))
radio_invest = float(input("Enter the investment in radio\n"))
newspaper_invest = float(input("Enter the investment in newspaper\n"))
New_Unemployment_Rate = 5.3
print ('Predicted Sales: \n', regr.predict([[TV_invest,radio_invest,newspaper_inves
```

```
Intercept: 
 2.9388893694594085
Coefficients: 
 [ 0.04576465  0.18853002 -0.00103749]
Enter the investment in TV
230
Enter the investment in radio
40
Enter the investment in newspaper
70
Predicted Sales: 
 [20.93333399]
```

In [26]:

```
X = sm.add_constant(X) # adding a constant

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results                            
==============================================================================
========
Dep. Variable:                  sales   R-squared:                       
0.897
Model:                            OLS   Adj. R-squared:                  
0.896
Method:                 Least Squares   F-statistic:                     
570.3
Date:                Sat, 22 May 2021   Prob (F-statistic):           
1.58e-96
Time:                        22:45:16   Log-Likelihood:                
-386.18
No. Observations:                 200   AIC:                             
780.4
Df Residuals:                     196   BIC:                             
793.6
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
========
                 coef    std err          t      P>|t|      [0.025      
0.975]
------------------------------------------------------------------------------
--------
const          2.9389      0.312      9.422      0.000       2.324       
3.554
TV             0.0458      0.001     32.809      0.000       0.043       
0.049
radio          0.1885      0.009     21.893      0.000       0.172       
0.206
newspaper     -0.0010      0.006     -0.177      0.860      -0.013       
0.011
==============================================================================
========
Omnibus:                       60.414   Durbin-Watson:                   
2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              
151.241
Skew:                          -1.327   Prob(JB):                     
1.44e-33
Kurtosis:                       6.332   Cond. No.                         
454.
==============================================================================
========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is 
correctly specified.
```

In [ ]:

```

```

F-statistic> Prob F-statistic

Therefore Null Hypothesis is rejected.

METHOD 2

ASSIGN_2_PROBLEM3

Predicting Value of Y given X1 and X2.

1.Enter the values of X1 ,X2 and X3

2. Get the predicted value of Y 3.Calculate the value of Root Mean Squared Error

In [27]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import statsmodels.api as sm
import numpy as np

df = pd.read_csv("/home/akash/Desktop/github/MACHINE LEARNING/data.csv")
X = df[['TV','radio','newspaper']]
y = df['sales']
regr = linear_model.LinearRegression()
regr.fit(X, y)
x1 = float(input("Enter the investment in TV to predict Sales\n"))
x2 = float(input("Enter the investment in radio to predict Sales\n"))
x3 = float(input("Enter the investment in newspaper to predict Sales\n"))
predictedy = regr.predict([[x1,x2,x3]])
print("Predicted value of Y for ",x1,", " ,x2,"and ",x2,"is",predictedy)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
df=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred
```

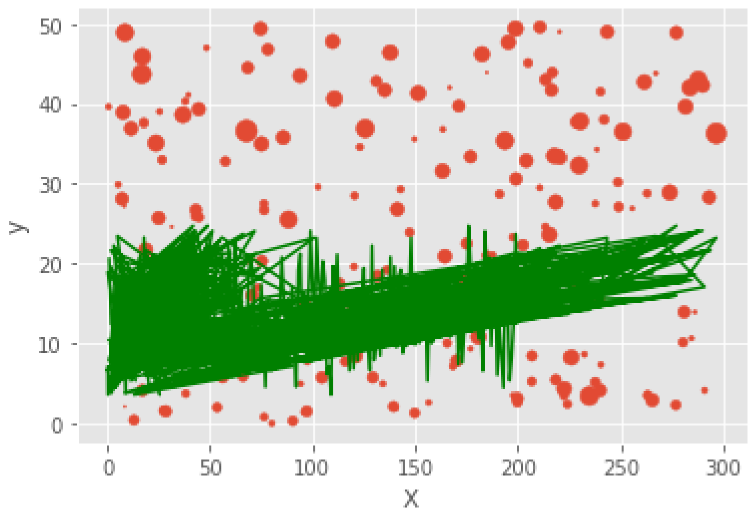
```
Enter the investment in TV to predict Sales
240
Enter the investment in radio to predict Sales
70
Enter the investment in newspaper to predict Sales
40
Predicted value of Y for  240.0 ,  70.0 and  70.0 is [27.07800574]
Mean Absolute Error: 0.975
Mean Squared Error: 2.0765
Root Mean Squared Error: 1.4410065926289164
```

In [28]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
dataset = pd.read_csv('/home/akash/Desktop/github/MACHINE LEARNING/data.csv')
dataset.head()
dataset.describe()

from sklearn.linear_model import LinearRegression
X = dataset.drop('sales', axis=1)
y = dataset['sales']
model = LinearRegression()
model.fit(X, y)
model = LinearRegression().fit(X, y)
r_sq = model.score(X, y)
print('coefficient of determination:', r_sq)
print('intercept value of the model is :', model.intercept_)
print('Slope is :', model.coef_)
y_pred = model.predict(X)
print('predicted response:', y_pred, sep='\n')
plt.scatter(X["TV"],X["radio"],X["newspaper"])
plt.plot(X, y_pred, color = "g")
# putting labels
plt.xlabel('X')
plt.ylabel('y')
# function to show plot
plt.show()
```

```
coefficient of determination: 0.8972508370448044
intercept value of the model is : 3.0052094200978523
Slope is : [-0.00057983  0.04577592  0.18838318 -0.00124333]
predicted response:
[20.57251457 12.38846299 12.35144103 17.64543251 13.24052459 12.5186
6879
 11.78301601 12.180724    3.78802522 12.60887762  7.08715353 17.3425
6434
 10.61745283  8.88300418 18.47999068 20.8606524  12.85204475 23.2667
1246
 10.00098857 14.2150162  18.14231423 14.79119255  6.52973904 16.5930
3652
  8.19342866 15.65971941 15.03489413 17.09729941 19.45415511  9.1829
9762
 21.6725338  11.38463112  7.68076947 18.91087004  7.62020678 17.0531
9819
 23.44631521 15.65194604  9.94172832 20.48118527 16.41271591 17.3270
7577
 21.63606281 14.00032963  8.91570435 15.19335755  8.9046642  21.7539
2107
 16.29149899  8.19695581 12.6626357   9.37496322 20.68673027 19.9629
0112
 20.40431955 21.30909879  8.54841699 12.80257742 21.91749556 18.1611
4677
  5.76901079 22.90645956 16.80885425 13.23498545 16.99561148  7.8763
2378
  9.03979324 12.06124085 18.99926536 21.1250438  17.7944357  10.6441
2377
 10.38232525  9.92057415 17.34824213 11.85594698  4.4950767  13.8272
4645
```

```
  8.82756268  9.69065949 11.45762825 14.66122134 10.18778083 14.4263
657
 20.78753503 15.1838158  11.60811119 15.5913576  11.70812929 16.9200
2973
 10.01166572  4.50260156 19.15418328 21.22197727 10.49027175 16.3118
9003
 12.64629333 15.34104735 24.11404042 16.93464073 13.87534134 23.2269
0723
 17.64796272 14.76413771 20.30310486 17.9238701   6.12285398  7.1084
0093
  3.58519686 19.69326737 14.75154529 21.13402399 13.88063488 16.4011
3163
 15.29164402 12.90360882 11.97142784  6.56637531 15.54311076  6.8100
8756
 14.39441514  7.82030899 13.62042032 15.07096712 19.43082627  9.1145
3063
 10.5538167   6.59078161 22.2416437   7.86508737 10.41045195 15.5612
9421
  8.43402145 19.243822   11.80609249 13.98077075 11.43301748 20.8241
0959
  9.74661397 19.65590883  9.46988128 18.36443847 19.22308369  8.7408
8909
 10.06448347  9.68965418 15.27515327 23.22779482 12.23534643  9.7990
9228
 18.33947913  9.97783056 16.33349025 18.18926965 15.47512722  5.2805
9543
 15.34441549  9.98551468 10.34373591 12.36435693 14.17976864 13.5170
9534
 14.91278707 17.31778078 11.03699433 14.17847989 10.78411871 13.3297
3051
 17.14157897 17.90963701  7.35722248 14.31389253  7.56747253 11.9345
9022
 13.7085193  24.73847813 19.93725072 12.11868528 15.97143167 12.3432
8192
 10.54824625 13.8849212   6.50856493 24.07488133 18.49113153 20.7531
3701
  9.6450546  17.02794864 18.60146081  6.00139443 12.43794837  8.3770
4275
  4.41373038 18.43576363 16.44377927  5.32006315  8.11608159 12.7367
4753
 23.70165968 15.12311317]
```



In [29]:

```
import statsmodels.api as sm
X = sm.add_constant(X) # adding a constant

model = sm.OLS(y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
====================================================================
==========
Dep. Variable:                  sales   R-squared:
0.897
Model:                            OLS   Adj. R-squared:
0.895
Method:                 Least Squares   F-statistic:
425.7
Date:                Sat, 22 May 2021   Prob (F-statistic):
3.94e-95
Time:                        22:46:29   Log-Likelihood:
-386.14
No. Observations:                 200   AIC:
782.3
Df Residuals:                     195   BIC:
798.8
Df Model:                           4
Covariance Type:            nonrobust
====================================================================
==========
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------
----------
const          3.0052      0.394      7.623      0.000       2.228
3.783
Unnamed: 0    -0.0006      0.002     -0.276      0.783      -0.005
0.004
TV             0.0458      0.001     32.725      0.000       0.043
0.049
radio          0.1884      0.009     21.784      0.000       0.171
0.205
newspaper     -0.0012      0.006     -0.210      0.834      -0.013
0.010
====================================================================
==========
Omnibus:                       60.267   Durbin-Watson:
2.085
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
150.423
Skew:                          -1.325   Prob(JB):
2.17e-33
Kurtosis:                       6.320   Cond. No.
653.
====================================================================
==========

Warnings:
```

```
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

F-statistic> Prob F-statistic Therefore Null Hypothesis is rejected.