Machine Learning Lab [01-07-2022]
Linear Regression Implementation
Nikhil Saraogi [21MCA1080]

Objective and Concept of linear Regression-

A linear regression analysis is a procedure to estimate a linear connection between one or more independent variables and the dependent variable. In the specimen of statistics and machine learning, it has been acknowledged as one of the most well-known and understood algorithms. This article cites the basic concept of linear regression. This includes the concept of math used for the implementation of an algorithm, also the use of it in the market.

Concept- Liner itself is defined as progression from one stage to another stage in a series of steps. This means it is used with continuous variables. But the drawback of linear regression is that it’s not good for the classification of different models.

If suppose we have an independent variable on X-axis and dependent variable on Y-axis. Let’s take the data point on the X-axis and Y-axis which is increasing. Here, we will get a positive linear regression line. Suppose we have dada variables that are increasing on X-axis and decreasing on Y-axis, then, in this case, we will catch a negative linear regression line. Once we add all the data points on the graph, the biggest task is to create the best fit line. After the line of regression is drawn using Y=MX+C, now the task is to predict the values. In this, the main goal is to reduce the error between the actual value and the estimated values, i.e. to reduce the distance between predicted or real value and search for the best fit line. The best fit line is the one that has the least error or a least distance between the actual value and the estimated value. So, in simple words, we must minimize the error.

For instance, we have speed on the X-axis and the distance covered on the Y-axis, with a time as constant. If suppose we draw a graph between the speed of travel and the distance covered by the vehicle in a fixed unit time, we will get a positive relation. Therefore, the equation of line we use here is Y=MX+C, where Y is the distance traveled in a fixed duration of time, X is the speed of a vehicle, M is a positive slope of a line and C is a Y-intercept of the line. In contrast to this, we can put this in another form by keeping the distance constant. Here, if we plot a graph between the speed of a vehicle and the time taken to travel of fixed distance, it will get a line with a negative relationship. For this, we have a negative slope of the line, i.e. Y= -MX+C, where y is time taken to travel a fixed distance, X is the speed of the vehicle, M is a negative slope of a line and finally C as y-intercept of line.

MATHEMATICAL IMPLEMENTATION

Let's take the value of X and Y respectively, finally solve the problem for linear regression step by step.

STEP 1: - Plot the value for X and Y,
STEP 2:-Calculate the mean for plotted points of the X and Y-axis.
STEP 3: - Calculate the equation for slope (M)
STEP 4: - After calculating the slope, head for calculating the constant variable (C)
STEP 5: - Once all the values for the equation Y=MX+C is known, calculate the values of Y for every independent values of X.

Formula for calculating the slope is $\mathbf{M} = \frac{\Sigma(\mathbf{X}-\bar{\mathbf{X}})\ (\mathbf{Y}-\bar{\mathbf{Y}})}{\Sigma\ (\mathbf{X}-\bar{\mathbf{X}})^2}$

Where, $(X-\bar{X})$ = Distance of all the points through the line Y

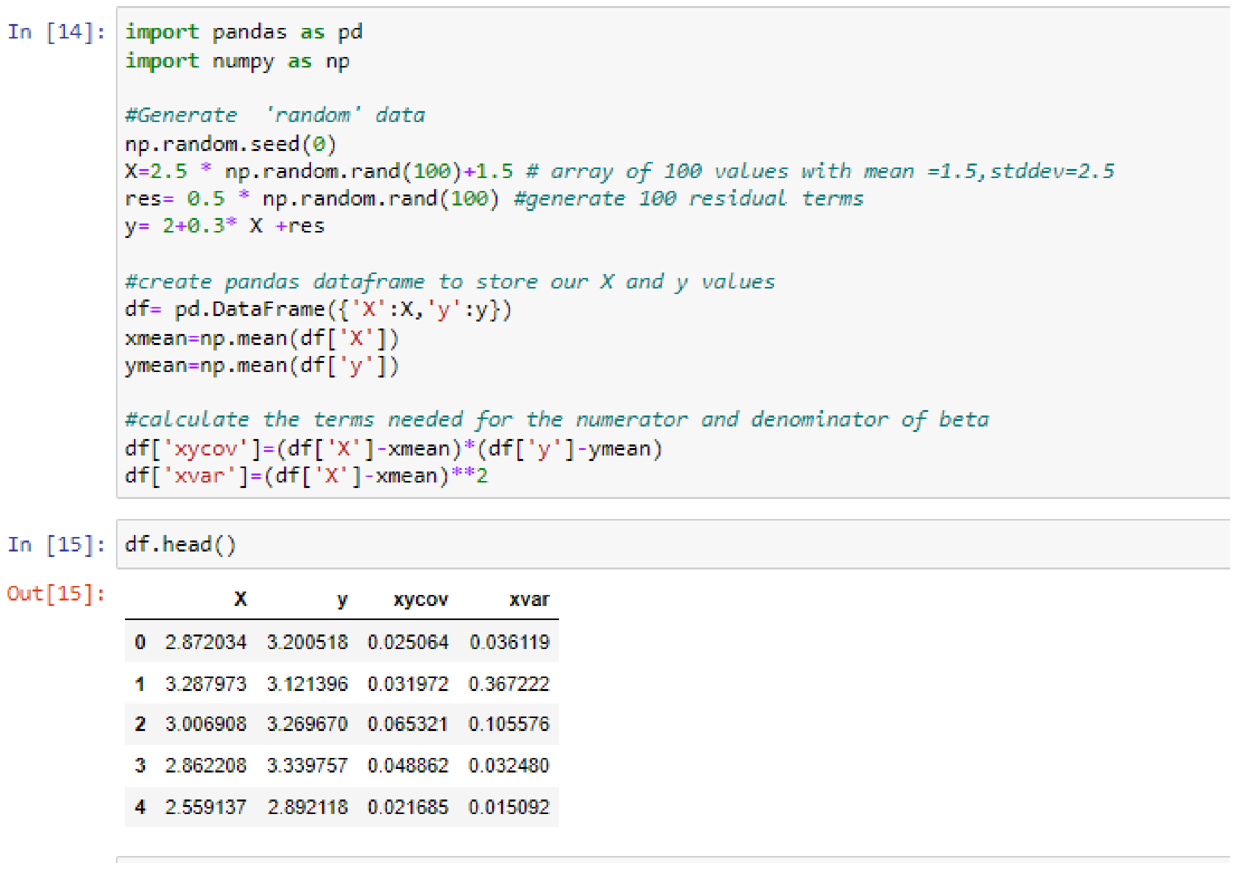
,$(Y-\bar{Y})$ = Distance of all the points from the line X

USE OF LINEAR REGRESSION

A. ANALYZING OF TRENDS AND SALES ESTIMATES
B. ESTIMATING THE IMPACT OF PRICE CHANGES
C. COSTING OF RISK FACTOR IN INSURANCE AND FINANCIAL SERVICES

Linear regression is a lengthy step of procedure, but it has a painless and an undemanding calculation to conduct. We can find different methods to calculate linear regression in different fields. Because of its simplicity, the use of a linear regression is intense in many sectors. A linear regression gives us a clear picture and provides a ground for selection criteria like data quality, classification and regression capabilities, comprehensible and transparent and computational complexity. This article depicts a brief information on the efficiency of a linear regression algorithm in the market.

1. Implementation from Scratch- (Random Values)



```
In [14]: import pandas as pd
         import numpy as np

         #Generate  'random' data
         np.random.seed(0)
         X=2.5 * np.random.rand(100)+1.5 # array of 100 values with mean =1.5,stddev=2.5
         res= 0.5 * np.random.rand(100) #generate 100 residual terms
         y= 2+0.3* X +res

         #create pandas dataframe to store our X and y values
         df= pd.DataFrame({'X':X,'y':y})
         xmean=np.mean(df['X'])
         ymean=np.mean(df['y'])

         #calculate the terms needed for the numerator and denominator of beta
         df['xycov']=(df['X']-xmean)*(df['y']-ymean)
         df['xvar']=(df['X']-xmean)**2
```

```
In [15]: df.head()
```

Out[15]:

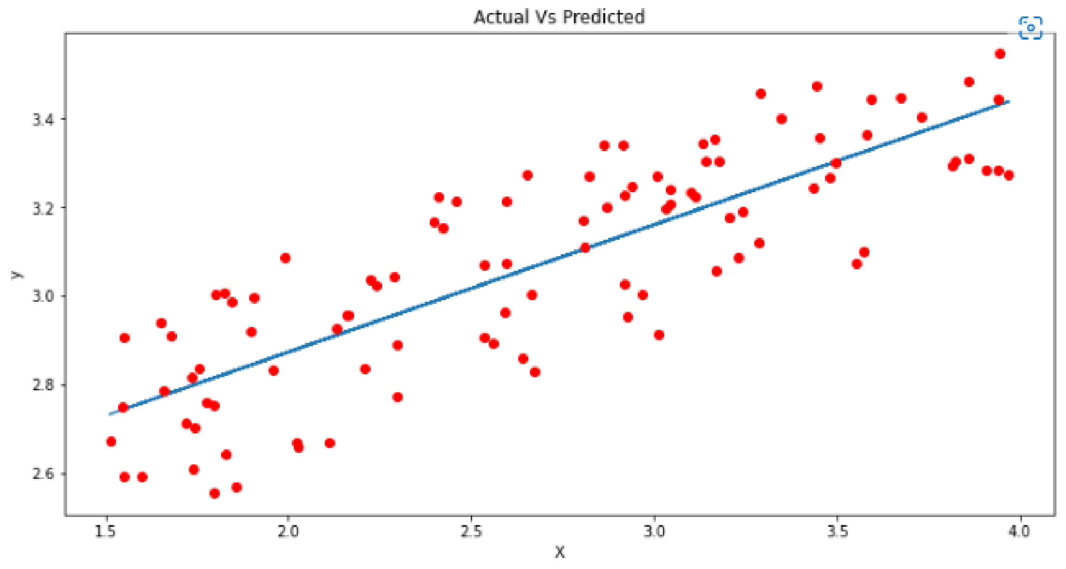
	X	y	xycov	xvar
0	2.872034	3.200518	0.025064	0.036119
1	3.287973	3.121396	0.031972	0.367222
2	3.006908	3.269670	0.065321	0.105576
3	2.862208	3.339757	0.048862	0.032480
4	2.559137	2.892118	0.021685	0.015092

In [11]:
```
#calculate beta and alpha
beta= df['xycov'].sum()/df['xvar'].sum()
alpha=ymean-(beta*xmean)
print(f'alpha={np.round(alpha,2)}')
print(f'beta={np.round(beta,2)}')
```

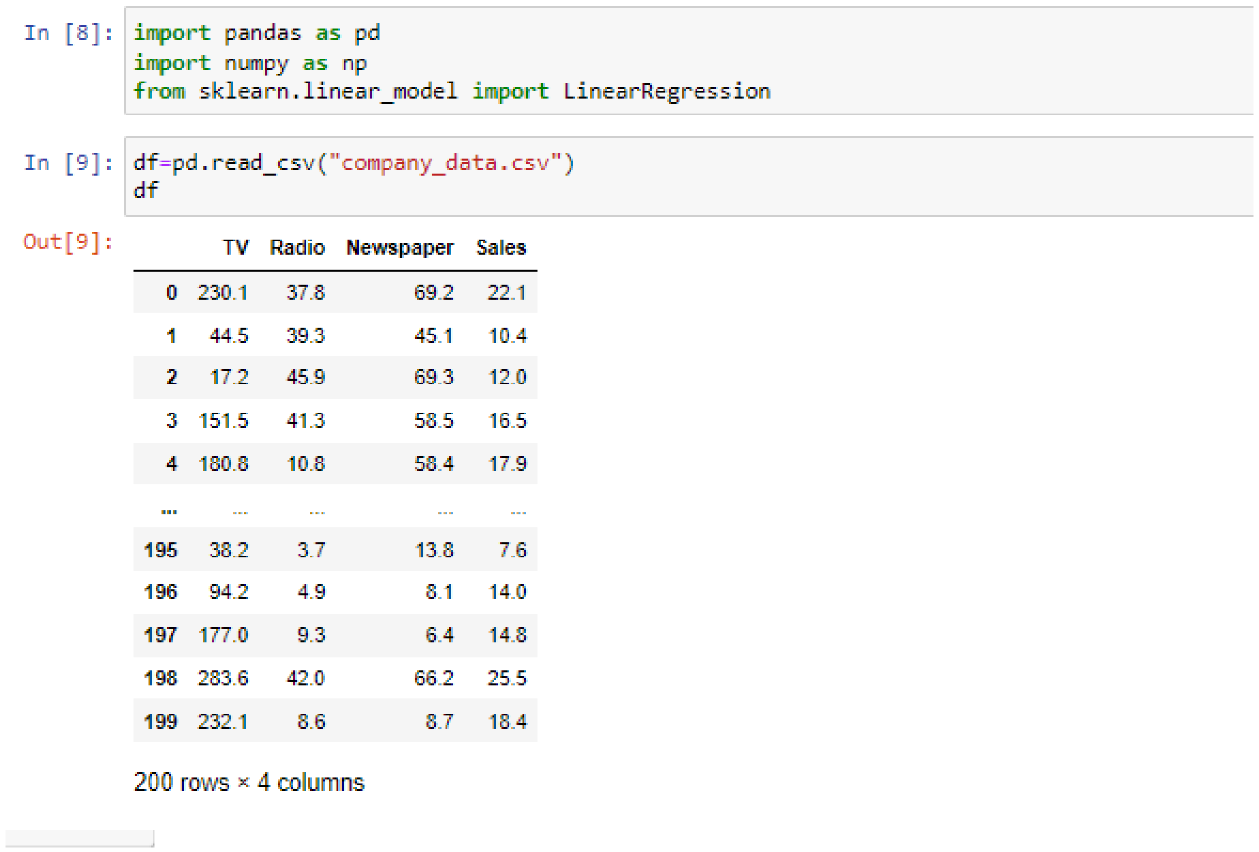
```
alpha=2.3
beta=0.29
```

In [12]:
```
ypred=alpha+beta*X
```

In [13]:
```
from matplotlib import pyplot as plt
%matplotlib inline
#plot regression against actual data
plt.figure(figsize=(12,6))
plt.plot(X,ypred) #regression line
plt.plot(X,y,'ro') #scatter plot showing actual data
plt.title('Actual Vs Predicted')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```



2. **Implementation using Scikit-learn (Company Dataset)**

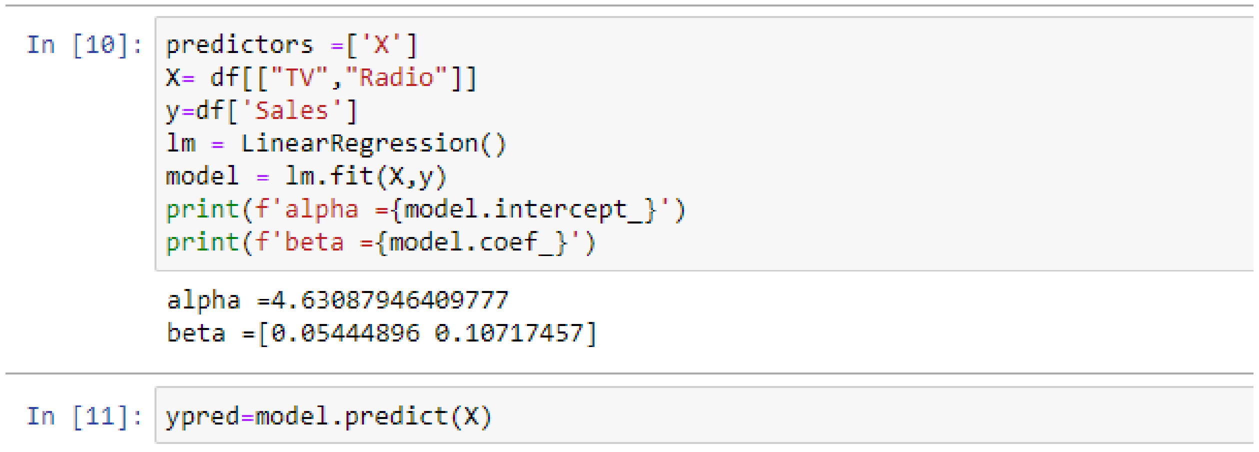


```
In [8]:  import pandas as pd
         import numpy as np
         from sklearn.linear_model import LinearRegression
```

```
In [9]:  df=pd.read_csv("company_data.csv")
         df

Out[9]:
            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   12.0
     3   151.5   41.3       58.5   16.5
     4   180.8   10.8       58.4   17.9
   ...     ...    ...        ...    ...
   195    38.2    3.7       13.8    7.6
   196    94.2    4.9        8.1   14.0
   197   177.0    9.3        6.4   14.8
   198   283.6   42.0       66.2   25.5
   199   232.1    8.6        8.7   18.4

   200 rows × 4 columns
```



```
In [10]: predictors =['X']
         X= df[["TV","Radio"]]
         y=df['Sales']
         lm = LinearRegression()
         model = lm.fit(X,y)
         print(f'alpha ={model.intercept_}')
         print(f'beta ={model.coef_}')

         alpha =4.63087946409777
         beta =[0.05444896 0.10717457]
```

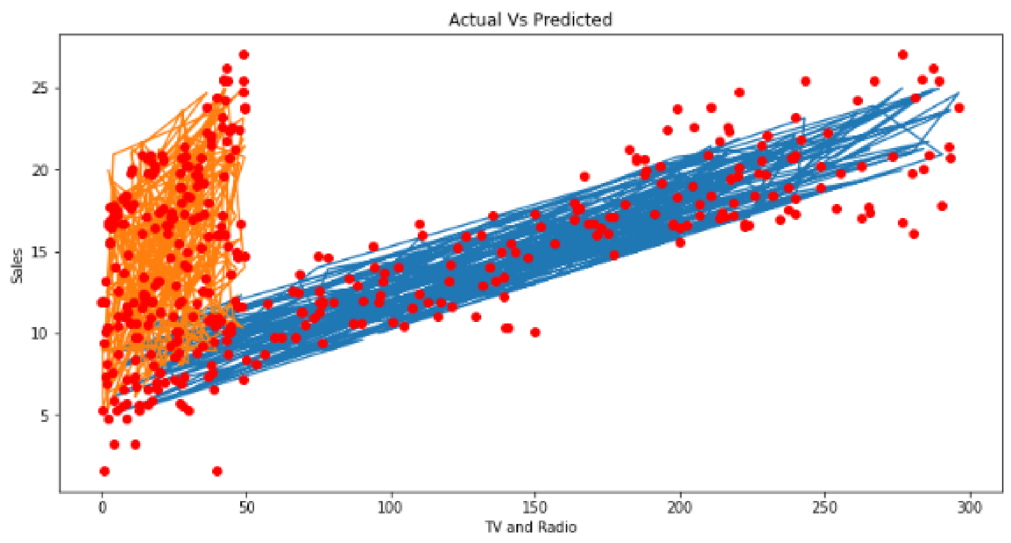
```
In [11]: ypred=model.predict(X)
In [12]: ypred
```

```
Out[12]: array([21.21078412, 11.26581887, 10.48671441, 17.30620681, 15.63273693,
                10.34542196, 11.27702065, 13.27626614,  5.32420713, 15.7884357 ,
                 8.85156828, 18.89326105,  9.68859218, 10.75417988, 19.26995575,
                20.38243344, 12.24510831, 24.19693004, 10.59582626, 15.21268363,
                19.49126811, 18.10365306,  7.05368143, 18.87282745,  9.37344932,
                19.32062224, 15.55185089, 19.49389028, 21.08221178, 10.18976923,
                23.61202043, 12.64300467, 10.0840803 , 21.23601486,  9.99168941,
                20.89860809, 23.85755331, 13.99264065,  9.83919073, 21.08572385,
                18.04678695, 17.8479762 , 23.58582996, 16.79663584,  8.75193486,
                16.57632034, 10.5759795 , 22.14092985, 18.69504157,  9.52745742,
                15.84202299, 11.12643101, 20.88281419, 19.52472489, 22.0212491 ,
                20.75520158,  8.03996233, 14.10457969, 21.42457912, 19.26492534,
                 7.75824801, 23.4347471 , 19.32172162, 13.39515504, 16.35620987,
                 9.38458127,  8.98251618, 13.76965098, 20.50436345, 21.14037783,
                18.75120943, 12.14197172,  9.62687247, 12.28747004, 18.88678214,
                10.23459567,  6.2997052 , 14.24645452,  8.12942354, 11.77220311,
                11.65234112, 18.12715599, 10.90653001, 13.1244568 , 20.86423915,
                17.12243079, 11.73263588, 15.00966701, 12.17167427, 15.73231986,
                12.46853029,  6.3488816 , 20.07476636, 22.20399557, 11.97914185,
                16.90911121, 15.7651051 , 16.9491583 , 24.93822776, 16.46155858,
                17.20117899, 24.65998836, 20.96994143, 16.70524181, 21.27670971,
                17.11229126,  7.17102377,  9.58521789,  5.38703068, 21.42014   ,
                17.80428628, 21.86382698, 15.84805026, 18.2511778 , 13.90455813,
                12.47110641, 13.74277117,  8.87651972, 15.42985551,  7.40198244,
                15.19679613,  7.98020812, 17.08466564, 15.0417867 , 20.58865461,
                10.64348878,  9.22467218,  8.99768611, 21.87753951,  9.16213238,
                 8.91310676, 19.38155011,  8.00339907, 20.18910917, 10.77698457,
                12.29796912, 10.20458114, 22.63090513,  9.74800617, 19.40345598,
                10.44940089, 18.97162298, 20.19507107, 10.93713581, 11.45505314,
                12.47370034, 18.48644931, 23.12442071, 11.0190752 ,  9.82985195,
                21.40442928, 12.11947011, 17.88716162, 18.21281692, 17.11777774,
                 6.09734523, 14.40573073, 12.92666072,  9.22267399, 13.7738197 ,
                15.96318493, 13.13400505, 16.82892341, 17.47730877, 12.58776386,
                17.7635543 ,  9.63527974, 16.44823231, 18.88850549, 21.24676946,
                 8.59655253, 15.82768205,  7.85228798, 14.56102391, 17.10472187,
                24.94863323, 21.39267336, 14.73405424, 19.94340841, 14.71937307,
```

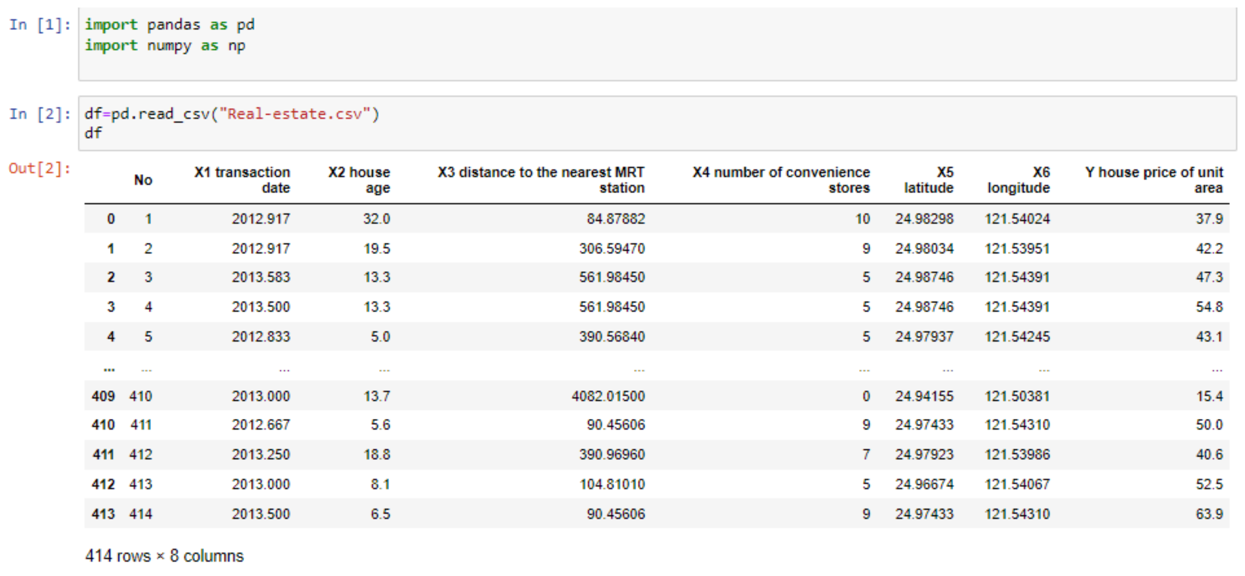
```
In [13]: new_X= [[300,200]] #predcting new values
         print(model.predict(new_X))

         [42.40048195]
```

```
In [15]: from matplotlib import pyplot as plt
         %matplotlib inline
         #plot regression against actual data
         plt.figure(figsize=(12,6))
         plt.plot(X,ypred) #regression line
         plt.plot(X,y,'ro') #scatter plot showing actual data
         plt.title('Actual Vs Predicted')
         plt.xlabel('TV and Radio')
         plt.ylabel('Sales')
         plt.show()
```



3. Implementation using Scikit-learn (Real-estate Dataset)



```
In [1]: import pandas as pd
        import numpy as np
```

```
In [2]: df=pd.read_csv("Real-estate.csv")
        df
```

Out[2]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...	...	...
409	410	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	411	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	412	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	413	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	414	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 8 columns

```
In [3]: df.columns
```

```
Out[3]: Index(['No', 'X1 transaction date', 'X2 house age',
               'X3 distance to the nearest MRT station',
               'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
               'Y house price of unit area'],
              dtype='object')
```

```
In [8]: #normalizing the data:
        from sklearn import preprocessing
        df_nor=preprocessing.normalize(df)
        df_nor=pd.DataFrame(df_nor)
        df_nor
```

Out[8]:

	0	1	2	3	4	5	6	7
0	0.000495	0.996909	0.015848	0.042037	0.004953	0.012373	0.060194	0.018770
1	0.000980	0.986502	0.009557	0.150258	0.004411	0.012243	0.059565	0.020682
2	0.001432	0.961229	0.006349	0.268276	0.002387	0.011928	0.058022	0.022580
3	0.001909	0.961141	0.006349	0.268263	0.002387	0.011928	0.058019	0.026159
4	0.002434	0.979673	0.002434	0.190095	0.002434	0.012158	0.059156	0.020977
...	...	...	...	...	...	...	...	...
409	0.089685	0.440334	0.002997	0.892920	0.000000	0.005456	0.026578	0.003369
410	0.199460	0.976753	0.002718	0.043899	0.004368	0.012120	0.058985	0.024265
411	0.196565	0.960519	0.008969	0.186531	0.003340	0.011918	0.057987	0.019370
412	0.200288	0.976223	0.003928	0.050829	0.002425	0.012108	0.058942	0.025460
413	0.200740	0.976305	0.003152	0.043860	0.004364	0.012110	0.058934	0.030984

414 rows × 8 columns

```
#splitting x and y:
X=df_nor.iloc[:,2:7]
X
```

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
0	0.015848	0.042037	0.004953	0.012373	0.060194
1	0.009557	0.150258	0.004411	0.012243	0.059565
2	0.006349	0.268276	0.002387	0.011928	0.058022
3	0.006349	0.268263	0.002387	0.011928	0.058019
4	0.002434	0.190095	0.002434	0.012158	0.059156
...	...	...	...	...	...
409	0.002997	0.892920	0.000000	0.005456	0.026578
410	0.002718	0.043899	0.004368	0.012120	0.058985
411	0.008969	0.186531	0.003340	0.011918	0.057987
412	0.003928	0.050829	0.002425	0.012108	0.058942
413	0.003152	0.043860	0.004364	0.012110	0.058934

414 rows × 5 columns

```
y=df_nor.iloc[:,7:8]
y
```

	Y house price of unit area
0	0.018770
1	0.020682
2	0.022580
3	0.026159
4	0.020977
...	...
409	0.003369
410	0.024265
411	0.019370
412	0.025460
413	0.030984

414 rows × 1 columns

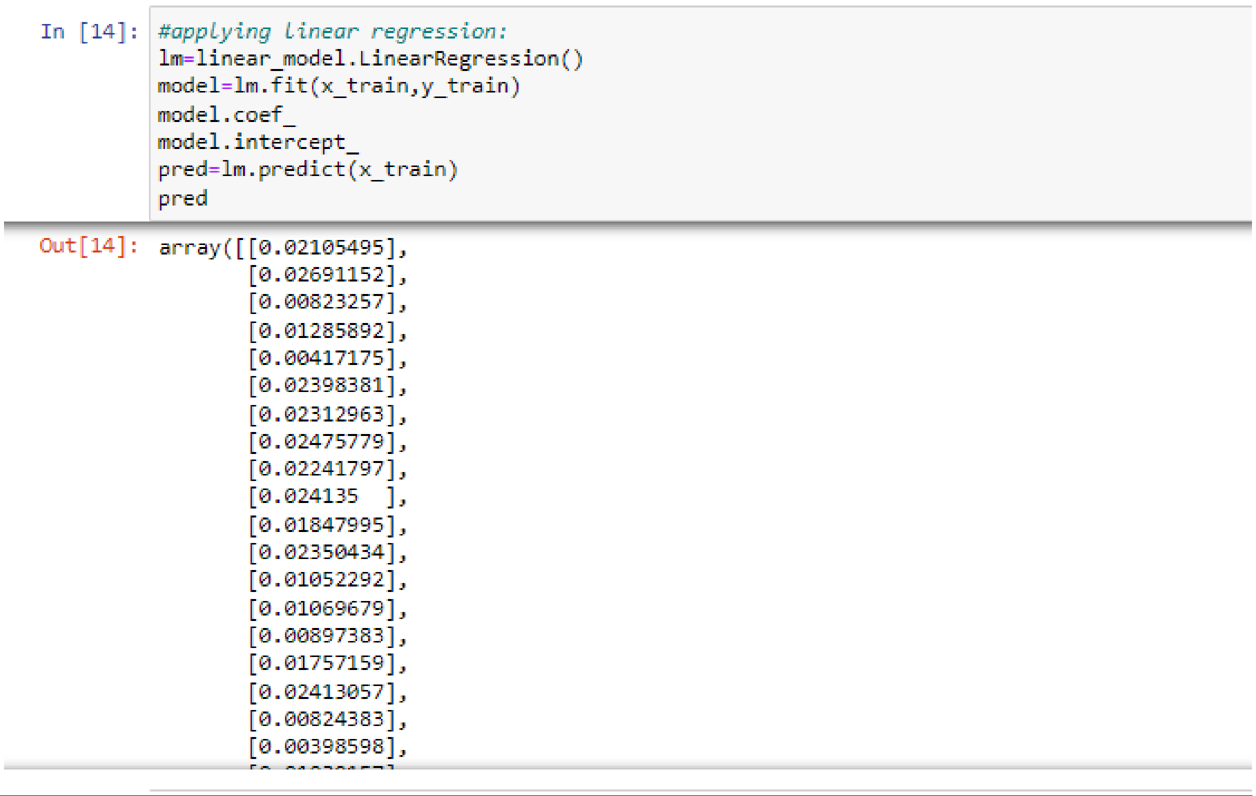
In [12]:
```
#splitting training and test data set:
from sklearn import model_selection
from sklearn import linear_model
```

In [13]:
```
x_train,x_test,y_train,y_test=model_selection.train_test_split(X,y,test_size=0.5)
x_train.shape
x_test.shape
y_train.shape
y_test.shape
x_train
```

Out[13]:

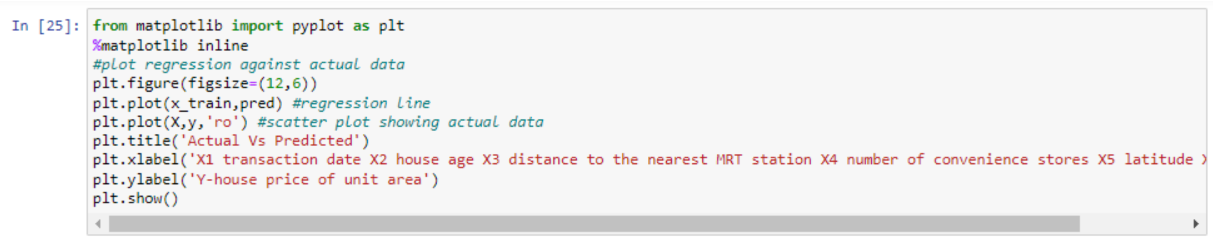
	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
27	0.005107	0.135752	0.002455	0.012255	0.059682
413	0.003152	0.043860	0.004364	0.012110	0.058934
186	0.007014	0.733319	0.001007	0.008378	0.040779
64	0.007775	0.442422	0.000000	0.011091	0.054005
155	0.003029	0.895993	0.000000	0.005475	0.026670
...	...	...	...	...	...
63	0.001245	0.255440	0.001915	0.011958	0.058200
85	0.000000	0.165567	0.004396	0.012196	0.059367
29	0.003434	0.218225	0.002418	0.012078	0.058781
167	0.013745	0.160890	0.003899	0.012173	0.059241
169	0.002979	0.696043	0.000355	0.008850	0.043109

207 rows × 5 columns

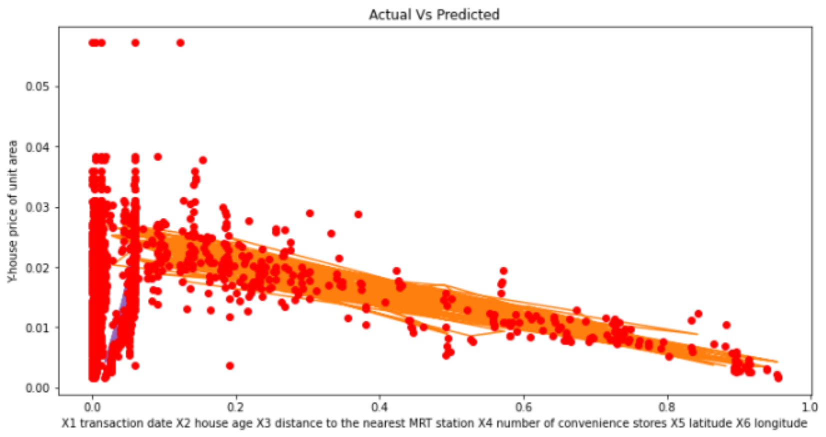


```
In [14]: #applying linear regression:
         lm=linear_model.LinearRegression()
         model=lm.fit(x_train,y_train)
         model.coef_
         model.intercept_
         pred=lm.predict(x_train)
         pred
```

```
Out[14]: array([[0.02105495],
                [0.02691152],
                [0.00823257],
                [0.01285892],
                [0.00417175],
                [0.02398381],
                [0.02312963],
                [0.02475779],
                [0.02241797],
                [0.024135  ],
                [0.01847995],
                [0.02350434],
                [0.01052292],
                [0.01069679],
                [0.00897383],
                [0.01757159],
                [0.02413057],
                [0.00824383],
                [0.00398598],
```



```
In [25]: from matplotlib import pyplot as plt
         %matplotlib inline
         #plot regression against actual data
         plt.figure(figsize=(12,6))
         plt.plot(x_train,pred) #regression line
         plt.plot(X,y,'ro') #scatter plot showing actual data
         plt.title('Actual Vs Predicted')
         plt.xlabel('X1 transaction date X2 house age X3 distance to the nearest MRT station X4 number of convenience stores X5 latitude
         plt.ylabel('Y-house price of unit area')
         plt.show()
```


Actual Vs Predicted
Y-house price of unit area
X1 transaction date X2 house age X3 distance to the nearest MRT station X4 number of convenience stores X5 latitude X6 longitude