Multiple Linear Regression

In multiple regression we have more than one independent variable and one dependent variable. The solution here will be represented by a plane and the generic form is given as below.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_k X_k + \varepsilon$$

Multiple linear regression attempts to model the relationship between two or more feartures and a response by fitting a linear equation to observed data.The steps to perform multiple linaer regression are almost similar to that of simple linear regression. The difference lies in the evaluation.You can use it to find out which factor has the highest impact on the predicted output and how different variables relate to each other.

In [2]:

```
## Import the libraries
import pandas as pd
import numpy as np

from sklearn import linear_model
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

In [3]:

```
## Import the dataset
data = datasets.load_boston()
df = pd.DataFrame(data.data, columns = data.feature_names)
df.head()
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7

In [4]:

```
target = pd.DataFrame(data.target,columns=['MEDV'])
target.head()
```

Out[4]:

	MEDV
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

Notice here, in following cell we are using more than one dependent variable in X

In [5]:

```
X= pd.DataFrame(df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAI
y = pd.DataFrame(target['MEDV'])
```

In [6]:

```
X.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7

In [7]:

```
#Split the data...
#split the data in 80/20 proportion
X_train ,X_test, y_train, y_test = train_test_split(X,y,test_size =0.2, random_sta
```

In [8]:

```
print('○ X : ',X.shape)
print("X_train :",X_train.shape)
print("X_test : ",X_test.shape)
print('\no y : ',y.shape)
print("y_train :",y_train.shape)
print("y_test: ",y_test.shape)
```

```
○ X :  (506, 13)
X_train : (404, 13)
X_test :  (102, 13)

○ y :  (506, 1)
y_train : (404, 1)
y_test:  (102, 1)
```

In [9]:

```
lm = linear_model.LinearRegression()
model = lm.fit(X_train,y_train)
print('coeficent',lm.coef_)
print('Intercept',lm.intercept_)
print('R square',lm.score(X_train,y_train))
```

```
coeficent [[-1.19443447e-01  4.47799511e-02  5.48526168e-03  2.3408
0361e+00
  -1.61236043e+01  3.70870901e+00 -3.12108178e-03 -1.38639737e+00
   2.44178327e-01 -1.09896366e-02 -1.04592119e+00  8.11010693e-03
  -4.92792725e-01]]
Intercept [38.09169493]
R square 0.7730135569264234
```

In [10]:

```
y_pred = lm.predict(X_test)
print('y_pred : ',y_pred[0:5]) ## Printing only first five y_pred elements
print(len(y_pred))
```

```
y_pred :  [[24.88963777]
 [23.72141085]
 [29.36499868]
 [12.12238621]
 [21.44382254]]
102
```

NOTE :
Having too many variables coluld potentially cause our model to become less accurate, especially if certain variables have no effect on the outcome or have a significant effect on the other variables. There are various methods to select the appropriate variable like:

1. Forward Selection
2. Backward Elimtion
3. Bi- directional Comparision

**please refer to notebook Feature Selection*

In [17]:

```
#importing the necessary libraries
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LinearRegression
# Sequential Forward Selection(sfs)
sfs = SFS(LinearRegression(), k_features=11, forward=True, floating=False, scoring

sfs.fit(X, y)
sfs.k_feature_names_
```

Out[17]:

```
('CRIM',
 'ZN',
 'CHAS',
 'NOX',
 'RM',
 'DIS',
 'RAD',
 'TAX',
 'PTRATIO',
 'B',
 'LSTAT')
```

In [12]:

```
#Performing Mutiple linear Regression on boston dataset using feature extracted fr
X = pd.DataFrame(df[['CRIM','ZN','CHAS','NOX','RM','DIS','RAD','TAX','PTRATIO','B
y = pd.DataFrame(target['MEDV'])
```

In [13]:

```
X.head()
```

Out[13]:

	CRIM	ZN	CHAS	NOX	RM	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	0.0	0.538	6.575	4.0900	1.0	296.0	15.3	396.90	4.9
1	0.02731	0.0	0.0	0.469	6.421	4.9671	2.0	242.0	17.8	396.90	9.1
2	0.02729	0.0	0.0	0.469	7.185	4.9671	2.0	242.0	17.8	392.83	4.0
3	0.03237	0.0	0.0	0.458	6.998	6.0622	3.0	222.0	18.7	394.63	2.9
4	0.06905	0.0	0.0	0.458	7.147	6.0622	3.0	222.0	18.7	396.90	5.3

In [14]:

```
y.head()
```

Out[14]:

	MEDV
0	24.0
1	21.6
2	34.7
3	33.4
4	36.2

In [15]:

```
from sklearn.model_selection import train_test_split
#split the data in 80/20 proportion
X_train ,X_test, y_train, y_test = train_test_split(X,y,test_size =0.2, random_sta
```

In [20]:

```
from sklearn import linear_model
lm = linear_model.LinearRegression()
model = lm.fit(X_train,y_train)
print('coeficent',lm.coef_)
print('Intercept',lm.intercept_)
print('R square',lm.score(X_train,y_train))
```

```
coeficent [[-1.19265889e-01  4.51412980e-02  2.34282671e+00 -1.6297
8884e+01
   3.68549285e+00 -1.37731203e+00  2.43212756e-01 -1.08465147e-02
  -1.04568509e+00  8.03588534e-03 -4.96639689e-01]]
Intercept [38.16377346]
R square 0.7729811407453248
```