VELLORE INSTITUTE OF TECHNOLOGY
CSE4020 Machine Learning
Lab Assessment - 2

17BCE0581

SATYAM SINGH CHAUHAN

Multiple Linear Regression

Importing the Required Libraries

- **matplotlib.pyplot**
- **pandas**
- **pylab**
- **numpy**

In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
%matplotlib inline
from sklearn import datasets
```

Load the diabetes dataset

In [2]:

```
diabetes = datasets.load_diabetes()
```

In [3]:

```
data1 = pd.DataFrame(data = np.c_[diabetes['data'],diabetes['target']],columns =
 diabetes['feature_names'] + ['target'])
```

Emploring and Studying the Datasets using *head* and *describe* functions

In [4]:

```
data1.head(10)
```

Out[4]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.(
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.(
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.(
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.(
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.(
5	-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076395	-0.(
6	-0.045472	0.050680	-0.047163	-0.015999	-0.040096	-0.024800	0.000779	-0.039493	-0.(
7	0.063504	0.050680	-0.001895	0.066630	0.090620	0.108914	0.022869	0.017703	-0.(
8	0.041708	0.050680	0.061696	-0.040099	-0.013953	0.006202	-0.028674	-0.002592	-0.(
9	-0.070900	-0.044642	0.039062	-0.033214	-0.012577	-0.034508	-0.024993	-0.002592	0.(

In [5]:

```
data1.describe()
```

Out[5]:

	age	sex	bmi	bp	s1	
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+
mean	-3.634285e-16	1.308343e-16	-8.045349e-16	1.281655e-16	-8.835316e-17	1.327024e-
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123996e-01	-1.267807e-01	-1.15613
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665645e-02	-3.424784e-02	-3.03584(
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670611e-03	-4.320866e-03	-3.81906!
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564384e-02	2.835801e-02	2.984439e-
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320442e-01	1.539137e-01	1.987880e-

Filtering the required Data

In [78]:

```
data2 = data1[['bmi','age','bp','s1','s5','s6','target']]
data3 = data1[['bmi','age','sex','bp','s1','s3','s5','s6','target']]
data4 = data1[['bmi','age','bp','s1','s3','s5','target']]
```

Exploring and Studying the Data2

In [56]:

```
data2.head()
```

Out[56]:

	bmi	age	bp	s1	s5	s6	target
0	0.061696	0.038076	0.021872	-0.044223	0.019908	-0.017646	151.0
1	-0.051474	-0.001882	-0.026328	-0.008449	-0.068330	-0.092204	75.0
2	0.044451	0.085299	-0.005671	-0.045599	0.002864	-0.025930	141.0
3	-0.011595	-0.089063	-0.036656	0.012191	0.022692	-0.009362	206.0
4	-0.036385	0.005383	0.021872	0.003935	-0.031991	-0.046641	135.0

Exploring and Studying the Data3

In [79]:

```
data3.head()
```

Out[79]:

	bmi	age	sex	bp	s1	s3	s5	s6	t
0	0.061696	0.038076	0.050680	0.021872	-0.044223	-0.043401	0.019908	-0.017646	
1	-0.051474	-0.001882	-0.044642	-0.026328	-0.008449	0.074412	-0.068330	-0.092204	
2	0.044451	0.085299	0.050680	-0.005671	-0.045599	-0.032356	0.002864	-0.025930	
3	-0.011595	-0.089063	-0.044642	-0.036656	0.012191	-0.036038	0.022692	-0.009362	
4	-0.036385	0.005383	-0.044642	0.021872	0.003935	0.008142	-0.031991	-0.046641	

Exploring and Studying the Data4

In [50]:

```
data4.head()
```

Out[50]:

	bmi	age	bp	s1	s3	s5	target
0	0.061696	0.038076	0.021872	-0.044223	-0.043401	0.019908	151.0
1	-0.051474	-0.001882	-0.026328	-0.008449	0.074412	-0.068330	75.0
2	0.044451	0.085299	-0.005671	-0.045599	-0.032356	0.002864	141.0
3	-0.011595	-0.089063	-0.036656	0.012191	-0.036038	0.022692	206.0
4	-0.036385	0.005383	0.021872	0.003935	0.008142	-0.031991	135.0

Split the data into training/testing sets

In [62]:

```
split2 = np.random.rand(len(data2)) < 0.8
train_data2 = data2[split2]
test_data2 = data2[~split2]
```

In [89]:

```
split3 = np.random.rand(len(data3)) < 0.9
train_data3 = data3[split3]
test_data3 = data3[~split3]
```

In [51]:

```
split4 = np.random.rand(len(data4)) < 0.8
train_data4 = data4[split4]
test_data4 = data4[~split4]
```

Create linear regression object
Train the model using the training sets
And Display The coefficients

In [63]:

```
from sklearn import linear_model
slr2 = linear_model.LinearRegression()
train_x2 = np.asanyarray(train_data2[['bmi','age','bp','s1','s5','s6']])
train_y2 = np.asanyarray(train_data2[["target"]])
slr2.fit(train_x2,train_y2)
print('Coefficient: ', slr2.coef_)
print('Intercept: ',slr2.intercept_)
```

```
Coefficient:  [[ 583.39946551 -102.30526358  333.3440783  -204.29365
95   616.64781099
    30.90767484]]
Intercept:  [152.97079742]
```

In [90]:

```
from sklearn import linear_model
slr3 = linear_model.LinearRegression()
train_x3 = np.asanyarray(train_data3[['bmi','age','sex','bp','s1','s3','s5','s6'
]])
train_y3 = np.asanyarray(train_data3[["target"]])
slr3.fit(train_x3,train_y3)
print('Coefficient: ', slr3.coef_)
print('Intercept: ',slr3.intercept_)
```

```
Coefficient:  [[ 526.11190048    9.96222165 -233.96335056  319.76490
89  -158.65303983
  -207.39184311  521.64182454  122.6196994 ]]
Intercept:  [150.78480642]
```

In [52]:

```
from sklearn import linear_model
slr4 = linear_model.LinearRegression()
train_x4 = np.asanyarray(train_data4[['bmi','age','bp','s1','s3','s5']])
train_y4 = np.asanyarray(train_data4[["target"]])
slr4.fit(train_x4,train_y4)
print('Coefficient: ', slr4.coef_)
print('Intercept: ',slr4.intercept_)
```

```
Coefficient:  [[ 481.00809091  -39.47417345  350.17229208 -148.41499
095 -136.16664934
   573.86448337]]
Intercept:  [152.03787327]
```

Model 1 Make predictions using the testing set

Data 2

In [64]:

```
from sklearn.metrics import r2_score, mean_squared_error
test_x2 = np.asanyarray(test_data2[['bmi','age','bp','s1','s5','s6']])
test_y2 = np.asanyarray(test_data2[["target"]])
test_result2 = slr2.predict(test_x2)
```

Calculating R2 Score for Data 2

In [65]:

```
r2Score_2 = r2_score(test_y2, test_result2)
print('R2 Score: ',r2Score_2)
```

```
R2 Score:  0.49244783525800795
```

Model 2 Make predictions using the testing set

Data 3

In [91]:

```
from sklearn.metrics import r2_score, mean_squared_error
test_x3 = np.asanyarray(test_data3[['bmi','age','sex','bp','s1','s3','s5','s6'
]])
test_y3 = np.asanyarray(test_data3[["target"]])
test_result3 = slr3.predict(test_x3)
```

Calculating R2 Score for Data 3

In [92]:

```
r2Score_3 = r2_score(test_y3, test_result3)
print('R2 Score: ',r2Score_3)
```

```
R2 Score:  0.5583133109733476
```

Model 3 Make predictions using the testing set

Data 4

In [53]:

```
from sklearn.metrics import r2_score, mean_squared_error
test_x4 = np.asanyarray(test_data4[['bmi','age','bp','s1','s3','s5']])
test_y4 = np.asanyarray(test_data4[["target"]])
test_result4 = slr4.predict(test_x4)
```

Calculating R2 Score for Data 4

In [54]:

```
r2Score_4 = r2_score(test_y4, test_result4)
print('R2 Score: ',r2Score_4)
```

```
R2 Score:  0.5089621667814918
```

Best R2 Score Observed : 0.5583133109733476 of Data 3

Model 2 when compared, found out to be the best was with the Data 3 and attributes ('bmi','age','sex','bp','s1','s3','s5','s6')

In [ ]:

```

```