VELLORE INSTITUTE OF TECHNOLOGY
CSE4020 Machine Learning
Lab Assessment - 1

17BCE0581

SATYAM SINGH CHAUHAN

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 [26]:

```
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 [5]:

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

Out[5]:

	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 [6]:

```
data1.describe()
```

Out[6]:

	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 [9]:

```
data2 = data1[['bmi','target']]
```

Exploring and Studying the Data

In [10]:

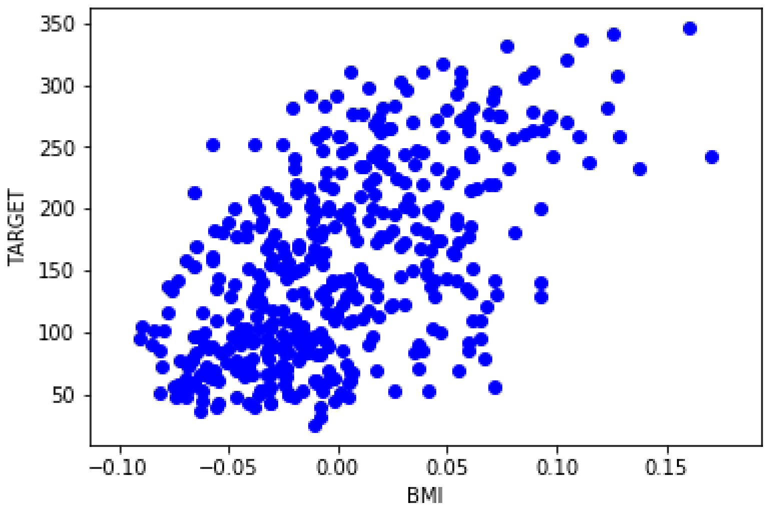
```
data2.head()
```

Out[10]:

	bmi	target
0	0.061696	151.0
1	-0.051474	75.0
2	0.044451	141.0
3	-0.011595	206.0
4	-0.036385	135.0

In [14]:

```
plt.scatter(data2.bmi,data2.target,color='blue')
plt.xlabel("BMI")
plt.ylabel("TARGET")
plt.show()
```



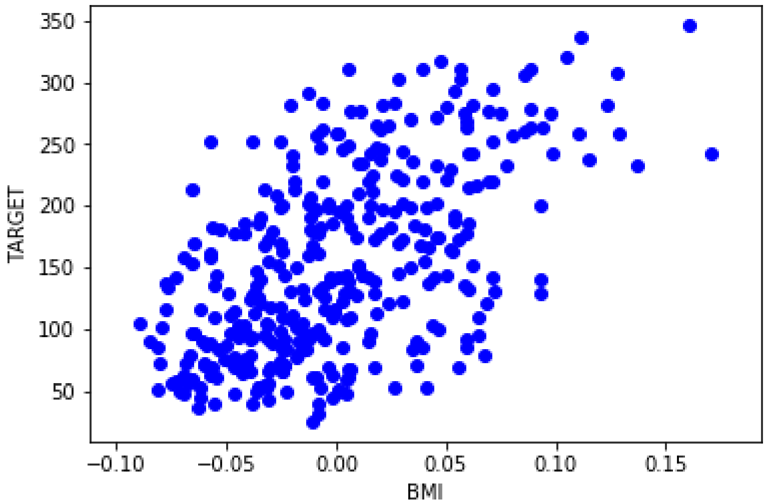
Split the data into training/testing sets

In [27]:

```
split = np.random.rand(len(data2)) < 0.8
train_data = data2[split]
test_data = data2[~split]
```

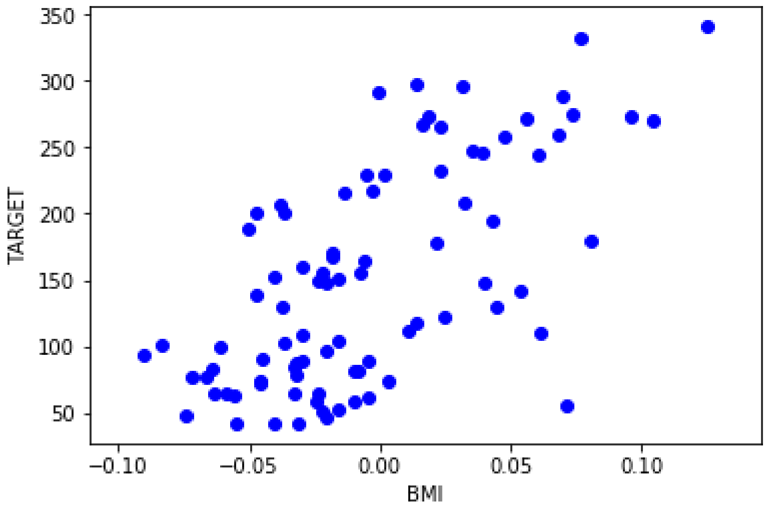
In [17]:

```
plt.scatter(train_data.bmi,train_data.target,color='blue')
plt.xlabel("BMI")
plt.ylabel("TARGET")
plt.show()
```



In [16]:

```
plt.scatter(test_data.bmi,test_data.target,color='blue')
plt.xlabel("BMI")
plt.ylabel("TARGET")
plt.show()
```



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

In [19]:

```
from sklearn import linear_model
slr = linear_model.LinearRegression()
train_x = np.asanyarray(train_data[["bmi"]])
train_y= np.asanyarray(train_data[["target"]])
slr.fit(train_x,train_y)
print('Coefficient: ', slr.coef_)
print('Intercept: ',slr.intercept_)
```

```
Coefficient:  [[900.59146152]]
Intercept:  [151.57092979]
```

Make predictions using the testing set

In [32]:

```
from sklearn.metrics import r2_score, mean_squared_error
test_x = np.asanyarray(test_data[["bmi"]])
test_y= np.asanyarray(test_data[["target"]])
test_result = slr.predict(test_x)
```

Calculating Mean Absolute Error

In [21]:

```
print("Mean Absolute Error: %.2f" %np.mean(np.absolute(test_result - test_y)))
```

```
Mean Absolute Error: 54.32
```

Calclulating Mean Squared Error

In [37]:

```
print('Mean squared error: %.2f' % mean_squared_error(test_y, test_result))
```

```
Mean squared error: 3684.43
```

Calculating Root Mean Squared Error

In [40]:

```
print('Root Mean squared error: %.2f'% np.sqrt(mean_squared_error(test_y, test_r
esult)))
```

```
Root Mean squared error: 60.70
```