In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
# from sklearn.datasets import load_boston  <-- It has been removed from sk
from sklearn.datasets import load_diabetes
```

In [3]:
```
diabetes = load_diabetes()
```

In [4]:
```
diabetes
```

Out[4]:
```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
          0.01990749, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06833155, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286131, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04688253,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
          0.04452873, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00422151,  0.00306441]]),
 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 31
0., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\n
```

```
Ten baseline variables, age, sex, body mass index, average blood\npressur
e, and six blood serum measurements were obtained for each of n =\n442 dia
betes patients, as well as the response of interest, a\nquantitative measu
re of disease progression one year after baseline.\n\n**Data Set Character
istics:**\n\n  :Number of Instances: 442\n\n  :Number of Attributes: First
10 columns are numeric predictive values\n\n  :Target: Column 11 is a quan
titative measure of disease progression one year after baseline\n\n  :Attr
ibute Information:\n      - age     age in years\n      - sex\n      - bmi
body mass index\n      - bp      average blood pressure\n      - s1      t
c, total serum cholesterol\n      - s2      ldl, low-density lipoproteins
\n      - s3      hdl, high-density lipoproteins\n      - s4      tch, tot
al cholesterol / HDL\n      - s5      ltg, possibly log of serum triglycer
ides level\n      - s6      glu, blood sugar level\n\nNote: Each of these
10 feature variables have been mean centered and scaled by the standard de
viation times the square root of `n_samples` (i.e. the sum of squares of e
ach column totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.edu/~boos/va
r.select/diabetes.html\n\nFor more information see:\nBradley Efron, Trevor
Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regressio
n," Annals of Statistics (with discussion), 407-499.\n(https://web.stanfor
d.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data_raw.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

In [5]:
```
df=pd.DataFrame(diabetes.data, columns = diabetes.feature_names)
target= pd.DataFrame(diabetes.target, columns=['Target'])
df = pd.concat([df, target], axis=1)
```

In [6]:
```
df.head()
```

Out[6]:

	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	0.019
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
age       0
sex       0
bmi       0
bp        0
s1        0
s2        0
s3        0
s4        0
s5        0
s6        0
Target    0
dtype: int64
```

Before Normalization

In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [9]:
```
X_train,X_test,y_train, y_test = train_test_split (df, target, test_size= 0
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[9]:
```
▾ LinearRegression
LinearRegression()
```

In [10]:
```
prediction = model.predict(X_test)
```

In [11]:
```
from sklearn.metrics import mean_squared_error , r2_score

mse = mean_squared_error(y_test, prediction)
r_squared = r2_score(y_test, prediction)
```

In [12]:
```
print(f"Mean Squared Error( MSE): {mse}")
print(f"R-squared : {r_squared}")
```

```
Mean Squared Error( MSE): 2.0977672827018797e-27
R-squared : 1.0
```

After Normalization

In [13]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
normalize_data = scaler.fit_transform(df)
normalize_df = pd.DataFrame(normalize_data, columns= df.columns)
```

In [14]:
```
normalize_df.head()
```

Out[14]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.800500	1.065488	1.297088	0.459841	-0.929746	-0.732065	-0.912451	-0.054499	0.418
1	-0.039567	-0.938537	-1.082180	-0.553505	-0.177624	-0.402886	1.564414	-0.830301	-1.436
2	1.793307	1.065488	0.934533	-0.119214	-0.958674	-0.718897	-0.680245	-0.054499	0.060
3	-1.872441	-0.938537	-0.243771	-0.770650	0.256292	0.525397	-0.757647	0.721302	0.476
4	0.113172	-0.938537	-0.764944	0.459841	0.082726	0.327890	0.171178	-0.054499	-0.672

In [15]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(normalize_df, target, t

normalized_model = LinearRegression()
normalized_model.fit(X_train,y_train)
```

Out[16]:

```
▾ LinearRegression
LinearRegression()
```

In [17]:
```
predictions =  normalized_model.predict(X_test)
```

In [18]:
```
normalized_mse = mean_squared_error(y_test, predictions)
normalized_Rsquared = r2_score(y_test, predictions)
```

In [19]:
```
print(f"Mean Squared Error( MSE): {mse}")
print(f"R-squared : {r_squared}")
```

```
Mean Squared Error( MSE): 2.0977672827018797e-27
R-squared : 1.0
```

In [20]:
```
print(f"Normalized Mean Squared Error : {normalized_mse}")
print(f"Normalized R Squared : {normalized_Rsquared}")
```

```
Normalized Mean Squared Error : 5.361843254758834e-27
Normalized R Squared : 1.0
```

In [ ]:

In [ ]: