In [179…
```
#Load all packages and libraries
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
```

In [181…
```
#Load dataset
diabetes = datasets.load_diabetes()
```

In [183…
```
diabetes
```

Out[183…
```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.0025922
6,
         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.,
310., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  4
9.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 34
1.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  9
2.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 18
2.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 16
3.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 17
0.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 13
4.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  9
2.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  8
1.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 20
0.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 15
8.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 23
```

```
5.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  7
1.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 21
4.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 12
7.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 13
7.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 12
9.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 15
5.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 18
5.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 22
0.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 17
7.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  9
1.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 11
6.,
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  6
6.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 23
3.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  8
9.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 17
2.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 10
9.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 13
5.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 21
6.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  7
1.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 32
1.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 16
8.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  6
9.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 25
8.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  7
2.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  5
5.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 31
0.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 13
2.,
       220.,  57.]),
```

```
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n
\nTen baseline variables, age, sex, body mass index, average blood\npres
sure, and six blood serum measurements were obtained for each of n =\n44
2 diabetes patients, as well as the response of interest, a\nquantitativ
e measure of disease progression one year after baseline.\n\n**Data Set
Characteristics:**\n\n:Number of Instances: 442\n\n:Number of Attribute
s: First 10 columns are numeric predictive values\n\n:Target: Column 11
is a quantitative measure of disease progression one year after baseline
\n\n:Attribute Information:\n    - age     age in years\n    - sex\n
- bmi     body mass index\n    - bp      average blood pressure\n    - s
1      tc, total serum cholesterol\n    - s2      ldl, low-density lipop
roteins\n    - s3      hdl, high-density lipoproteins\n    - s4      tc
h, total cholesterol / HDL\n    - s5      ltg, possibly log of serum tri
glycerides level\n    - s6      glu, blood sugar level\n\nNote: Each of
these 10 feature variables have been mean centered and scaled by the sta
ndard deviation times the square root of `n_samples` (i.e. the sum of sq
uares of each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncsu.e
du/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley
Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.\n(ht
tps://web.stanford.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 [185…
```python
# Show feature names
print(diabetes.feature_names)
```

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

In [187…
```python
# Create Feature and Target variables
X = diabetes.data
y = diabetes.target
print("dimension of X:", X.shape)
print("dimension of y:", y.shape)
```

```
dimension of X: (442, 10)
dimension of y: (442,)
```

In [189…
```python
# Split data into testing & training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
```

In [191…
```python
# Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
```

In [193…
```
# Dimensions of training data and testing data
print("feature training data:", X_train.shape, "/ target training data:",
print("feature testing data:", X_test.shape, "/ target testing data:" , y
print("scaled feature training data:", X_train_scaled.shape)
print("scaled feature test data:", X_test_scaled.shape)
```

```
feature training data: (353, 10) / target training data: (353,)
feature testing data: (89, 10) / target testing data: (89,)
scaled feature training data: (353, 10)
scaled feature test data: (89, 10)
```

In [195…
```
#BUILDING LINEAR REGRESSION MODEL
# 1. defining the model
model = linear_model.LinearRegression()
```

In [197…
```
# 2. build training model
model.fit(X_train_scaled, y_train)
```

Out[197…

▾ LinearRegression

```
LinearRegression()
```

In [199…
```
# Applying training model to testing dataset
Y_pred = model.predict(X_test_scaled)
print(Y_pred)
```

```
[158.82689128 232.78373635  87.28268709 128.95236713  67.66313843
 126.14110688 219.14585502 180.36516868 240.47185892 146.29544617
 218.07251394 227.55217003 234.06388096 156.16293495  52.44746772
 103.96782622  79.18036892 163.03148115 120.28946439 134.50558252
 183.04263076 186.44995146 141.97441238 163.88064257 175.35236569
 136.16146038 169.88077624 266.04171109 123.79984609 286.29077474
 180.62640457 144.61102647 172.07594755  79.99445059 112.84725673
 110.49595288 131.45259736 120.21140734 224.35302993  60.65423612
 166.51559644 198.70405821  81.47531693 120.5324054  198.83101816
 211.49369722 152.48611182 253.82678073 183.78230134 189.40414745
 169.22566031 151.17807948 166.9624241  164.53463363 172.14749675
 227.36590659 107.39736761  70.91704662  83.66134184 168.75349799
 131.92010357 145.88116863 178.11036137 129.13969108  77.21187989
 252.63863297 165.46166855 183.18564755 131.55842155 220.76088107
 159.7281096   57.15148271 132.12656623 211.1858594  189.50235046
 155.09555932 180.57209118 204.17449123 164.20392112 139.46522389
  88.3872361   66.50617918 162.8997838  125.95341377  93.95661619
 204.27347681  92.49964795 239.03740687  81.38544078]
```

In [201…
```
# PREDICTION RESULTS
# PRINT MODEL PERFORMANCE
print("Parameters:", model.coef_)       #coefficients of feature varibale
print("Intercept:", model.intercept_)
print("MSE: %.2f"
      % mean_squared_error(y_test, Y_pred))
print("Coefficient of determination:%.2f"
```

```
      % r2_score(y_test, Y_pred))
```

```
Parameters: [  2.68557884  -9.93762507  19.923154    15.19270396 -41.56270
731
  26.38957583   4.12720436   6.41289538  37.92535719   5.31541486]
Intercept: 153.44759206798867
MSE: 3410.87
Coefficient of determination:0.49
```

In [203…
```
feature_names = diabetes.feature_names
print(feature_names[0])
```

```
age
```

In [97]:
```
# Each model coeffecient is multiplied by its corresponding feature varia
# The sum of all products of model coef and features should produce the v
# Y^ = model.coef_[0]*feature_names[0] +  model.coef_[1]*feature_names[1]
# model.coef_[10]*feature_names[10] = model.intercept_
```

In [205…
```
y_test
```

Out[205…
```
array([200., 321.,  53.,  89., 158.,  71., 257., 262., 306., 116., 237.,
       268., 246., 109.,  48., 118.,  59., 237.,  64., 187., 144., 217.,
       302., 154.,  85.,  42.,  58., 220.,  78., 270., 107.,  88., 174.,
        48.,  71., 142.,  83., 150., 248.,  70., 131., 292.,  37.,  64.,
       129., 202., 200., 308.,  90., 142., 216., 129., 151., 168., 252.,
       217.,  68., 134., 201., 147.,  71.,  25.,  70.,  51.,  77., 346.,
       200., 167.,  65., 180., 104.,  39.,  84., 249., 124.,  81.,  91.,
       121., 259., 168.,  55.,  75., 155.,  53.,  64., 265., 115., 215.,
        72.])
```

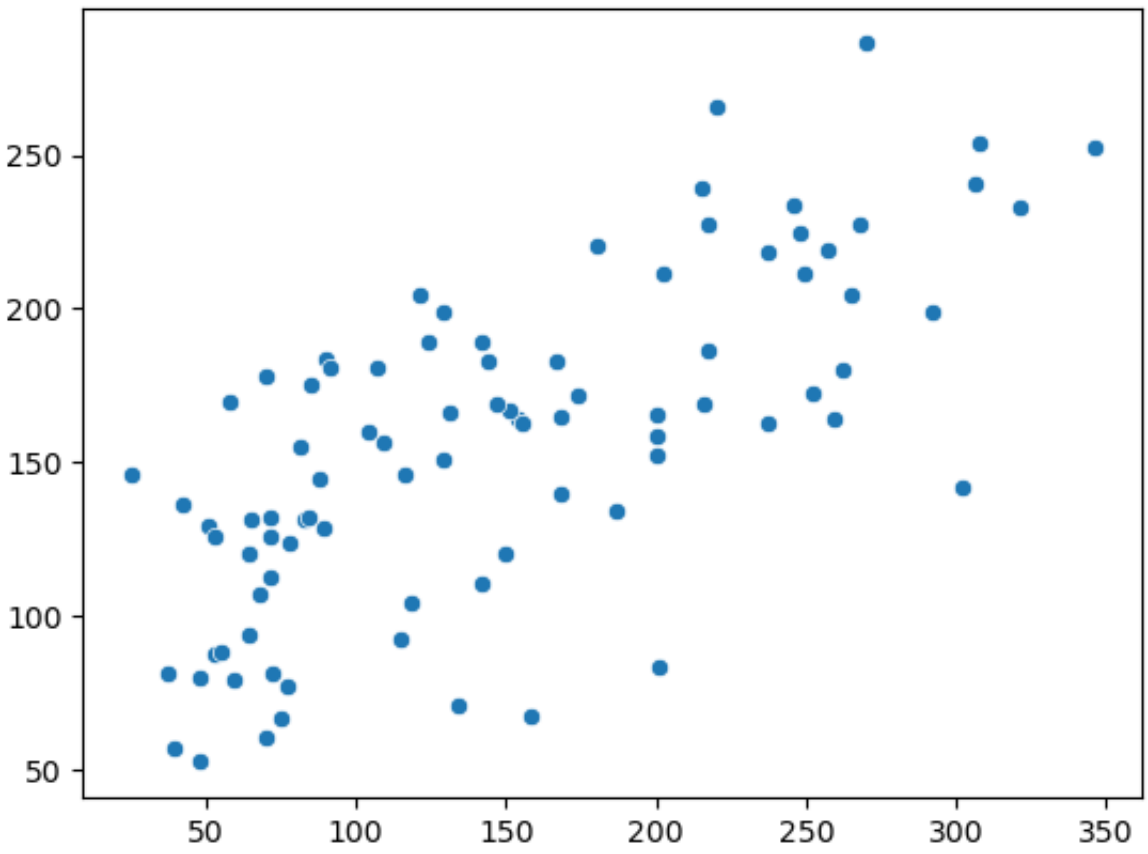
In [207…
```
Y_pred
```

Out[207…
```
array([158.82689128, 232.78373635,  87.28268709, 128.95236713,
        67.66313843, 126.14110688, 219.14585502, 180.36516868,
       240.47185892, 146.29544617, 218.07251394, 227.55217003,
       234.06388096, 156.16293495,  52.44746772, 103.96782622,
        79.18036892, 163.03148115, 120.28946439, 134.50558252,
       183.04263076, 186.44995146, 141.97441238, 163.88064257,
       175.35236569, 136.16146038, 169.88077624, 266.04171109,
       123.79984609, 286.29077474, 180.62640457, 144.61102647,
       172.07594755,  79.99445059, 112.84725673, 110.49595288,
       131.45259736, 120.21140734, 224.35302993,  60.65423612,
       166.51559644, 198.70405821,  81.47531693, 120.5324054 ,
       198.83101816, 211.49369722, 152.48611182, 253.82678073,
       183.78230134, 189.40414745, 169.22566031, 151.17807948,
       166.9624241 , 164.53463363, 172.14749675, 227.36590659,
       107.39736761,  70.91704662,  83.66134184, 168.75349799,
       131.92010357, 145.88116863, 178.11036137, 129.13969108,
        77.21187989, 252.63863297, 165.46166855, 183.18564755,
       131.55842155, 220.76088107, 159.7281096 ,  57.15148271,
       132.12656623, 211.1858594 , 189.50235046, 155.09555932,
       180.57209118, 204.17449123, 164.20392112, 139.46522389,
        88.3872361 ,  66.50617918, 162.8997838 , 125.95341377,
        93.95661619, 204.27347681,  92.49964795, 239.03740687,
        81.38544078])
```

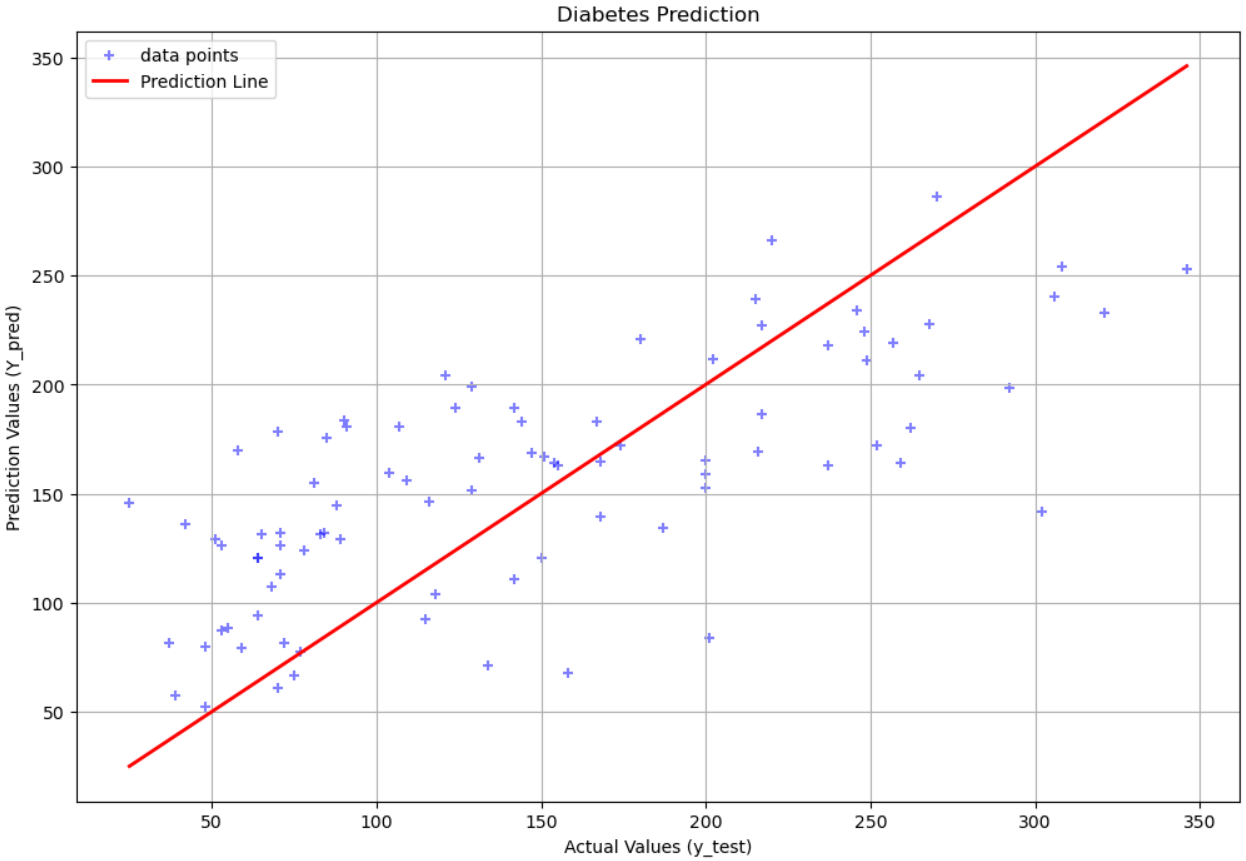
In [209…
```
# VISUALIZATIONS - SCATTERPLOTS
import seaborn as sns
import matplotlib.pyplot as plt
```

In [211…
```
# first look at data points on the plot
sns.scatterplot(x=y_test,y=Y_pred, )
```

Out[211… <Axes: >



In [213…
```
# FINAL & MAIN SCATTER PLOT WITH REGRESSION/PREDICTION LINE
plt.figure(figsize = (12, 8))
plt.scatter(y_test, Y_pred, marker ='+', alpha =0.5, color='blue', label
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         'r-',
         lw = 2,
         label = 'Prediction Line')
plt.title('Diabetes Prediction')
plt.xlabel('Actual Values (y_test)')
plt.ylabel('Prediction Values (Y_pred)')
plt.legend()
plt.grid(True)
plt.savefig("Diabetes Prediction(scaled)", dpi=300, bbox_inches = 'tight'
plt.show()
```


Diabetes Prediction
data points
Prediction Line
Prediction Values (Y_pred)
Actual Values (y_test)
350
300
250
200
150
100
50
50
100
150
200
250
300
350

In [ ]: