Predicting Disease Progression using Linear Regression

Name: Jaamie Maarsh Joy Martin

CS 6140 Machine Learning - Assignment 1

Few of the syntax has been adopted from Prof. Shanu Sushmitha and the RFE syntax has been referenced from google

```
import numpy as np
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split, cross_val_score,
KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

# Importing the inbuilt dataset from sklearn
from sklearn.datasets import load_diabetes
diabetes_df = load_diabetes()
print(diabetes_df.DESCR)

.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n
=
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression
one year after baseline

:Attribute Information:
    - age     age in years
    - sex
    - bmi     body mass index
    - bp      average blood pressure
```

```
    - s1      tc, total serum cholesterol
    - s2      ldl, low-density lipoproteins
    - s3      hdl, high-density lipoproteins
    - s4      tch, total cholesterol / HDL
    - s5      ltg, possibly log of serum triglycerides level
    - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and
scaled by the standard deviation times the square root of `n_samples`
(i.e. the sum of squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani
(2004) "Least Angle Regression," Annals of Statistics (with
discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)


# checking out column names
print(diabetes_df.feature_names)

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

print(diabetes_df)

{'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., 310., 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\nTen baseline variables, age,
sex, body mass index, average blood\npressure, and six blood serum
measurements were obtained for each of n =\n442 diabetes patients, as
well as the response of interest, a\nquantitative measure of disease
progression one year after baseline.\n\n**Data Set Characteristics:**\
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
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    -
s1      tc, total serum cholesterol\n    - s2      ldl, low-density
lipoproteins\n    - s3      hdl, high-density lipoproteins\n    - s4
tch, total cholesterol / HDL\n    - s5      ltg, possibly log of serum
triglycerides level\n    - s6      glu, blood sugar level\n\nNote:
Each of these 10 feature variables have been mean centered and scaled
by the standard deviation times the square root of `n_samples` (i.e.
the sum of squares of each column totals 1).\n\nSource
URL:\nhttps://www4.stat.ncsu.edu/~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(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_200
2.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'}
```

From the above it is found that it is not a regular dataframe as it is in the form of multi-dimensional arrays or nested lists. The input to the dataframe should be either a dictionary, a 1D or a 2D array.

```
# Convert to pandas DataFrame for easier manipulation
data = pd.DataFrame(data=diabetes_df.data,
columns=diabetes_df.feature_names)
target = pd.DataFrame(data=diabetes_df.target, columns=['target'])

diabetes_df_final = pd.concat([data, target], axis=1)
display(diabetes_df_final)
```

```
          age       sex       bmi        bp        s1        s2
s3  \
0    0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -
0.043401
1   -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163
0.074412
2    0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -
0.032356
3   -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -
0.036038
4    0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596
0.008142
..        ...       ...       ...       ...       ...       ...
...
437  0.041708  0.050680  0.019662  0.059744 -0.005697 -0.002566 -
0.028674
438 -0.005515  0.050680 -0.015906 -0.067642  0.049341  0.079165 -
0.028674
439  0.041708  0.050680 -0.015906  0.017293 -0.037344 -0.013840 -
0.024993
440 -0.045472 -0.044642  0.039062  0.001215  0.016318  0.015283 -
0.028674
441 -0.045472 -0.044642 -0.073030 -0.081413  0.083740  0.027809
0.173816

           s4        s5        s6  target
0   -0.002592  0.019907 -0.017646   151.0
1   -0.039493 -0.068332 -0.092204    75.0
2   -0.002592  0.002861 -0.025930   141.0
3    0.034309  0.022688 -0.009362   206.0
4   -0.002592 -0.031988 -0.046641   135.0
..        ...       ...       ...     ...
437 -0.002592  0.031193  0.007207   178.0
438  0.034309 -0.018114  0.044485   104.0
439 -0.011080 -0.046883  0.015491   132.0
440  0.026560  0.044529 -0.025930   220.0
441 -0.039493 -0.004222  0.003064    57.0
```

```
[442 rows x 11 columns]
```

```
# Information about the dataset
diabetes_df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   age     442 non-null    float64
 1   sex     442 non-null    float64
 2   bmi     442 non-null    float64
 3   bp      442 non-null    float64
 4   s1      442 non-null    float64
 5   s2      442 non-null    float64
 6   s3      442 non-null    float64
 7   s4      442 non-null    float64
 8   s5      442 non-null    float64
 9   s6      442 non-null    float64
 10  target  442 non-null    float64
dtypes: float64(11)
memory usage: 38.1 KB
```

It can be found that the dataset is free from null values. Further preprocessing can be done at this point.

Feature selection

Correlation matrix is an indicator to figure out the attributes which have high correlation with the target variable.

```
# Create a correlation matrix with plotly
correlation_matrix = diabetes_df_final.corr()

fig = px.imshow(correlation_matrix,
                text_auto=True,
                aspect="auto",
                color_continuous_scale='RdBu_r',
                title='Correlation Matrix for the Prediction of
Disease Progression')

fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"coloraxis":"coloraxis","hovertemplate":"x: %{x}<br>y: %
{y}<br>color: %{z}<extra></extra>","name":"0","texttemplate":"%
{z}","type":"heatmap","x":
["age","sex","bmi","bp","s1","s2","s3","s4","s5","s6","target"],"xaxis
```

```
":"x","y":
["age","sex","bmi","bp","s1","s2","s3","s4","s5","s6","target"],"yaxis
":"y","z":
[[1,0.17373710056366062,0.18508466614655544,0.3354275870670726,0.26006
08201502613,0.219243139847508,-7.518097487514619e-
2,0.2038408997287552,0.27077424141816636,0.30173100763283783,0.1878887
5071891967],[0.17373710056366062,1,8.816139902276222e-
2,0.2410104866490488,3.527681917552949e-2,0.1426372570335002,-
0.3790896292273318,0.3321150930829645,0.149916136495838,0.208133216200
38897,4.3061998451605354e-2],[0.18508466614655544,8.816139902276222e-
2,1,0.3954108987177125,0.24977742174241246,0.2611699111644256,-
0.3668109784050295,0.4138066018314402,0.44615653857325194,0.3886799939
0003893,0.5864501344746884],
[0.3354275870670726,0.2410104866490488,0.3954108987177125,1,0.24246402
270704165,0.18554846261290037,-
0.17876163122564298,0.257650053283515,0.39348010904483205,0.3904300231
16037,0.44148175856257077],[0.2600608201502613,3.527681917552949e-
2,0.24977742174241246,0.24246402270704165,1,0.8966629578104892,5.15193
64314820425e-
2,0.5422072805232409,0.5155029243689457,0.325716753060709,0.2120224810
1455062],
[0.219243139847508,0.1426372570335002,0.2611699111644256,0.18554846261
290037,0.8966629578104892,1,-
0.1964551237441792,0.6598168886666438,0.3183566651415615,0.29060037549
70445,0.17405358696874246],[-7.518097487514619e-2,-
0.3790896292273318,-0.3668109784050295,-
0.17876163122564298,5.1519364314820425e-2,-0.1964551237441792,1,-
0.7384927292583822,-0.39857729342870957,-0.2736973014758417,-
0.39478925067091825],
[0.2038408997287552,0.3321150930829645,0.4138066018314402,0.2576500532
83515,0.5422072805232409,0.6598168886666438,-
0.7384927292583822,1,0.617858973993728,0.4172121137122002,0.4304528847
447728],
[0.27077424141816636,0.149916136495838,0.44615653857325194,0.393480109
04483205,0.5155029243689457,0.3183566651415615,-
0.39857729342870957,0.617858973993728,1,0.4646688466913681,0.565882592
4427436],
[0.30173100763283783,0.20813321620038897,0.38867999390003893,0.3904300
23116037,0.325716753060709,0.2906003754970445,-
0.2736973014758417,0.4172121137122002,0.4646688466913681,1,0.382483484
2485807],[0.18788875071891967,4.3061998451605354e-
2,0.5864501344746884,0.44148175856257077,0.21202248101455062,0.1740535
8696874246,-
0.39478925067091825,0.4304528847447728,0.5658825924427436,0.3824834842
485807,1]]}],"layout":{"coloraxis":{"colorscale":[[0,"rgb(5,48,97)"],
[0.1,"rgb(33,102,172)"],[0.2,"rgb(67,147,195)"],
[0.3,"rgb(146,197,222)"],[0.4,"rgb(209,229,240)"],
[0.5,"rgb(247,247,247)"],[0.6,"rgb(253,219,199)"],
[0.7,"rgb(244,165,130)"],[0.8,"rgb(214,96,77)"],
```

```
[0.9,"rgb(178,24,43)"],[1,"rgb(103,0,31)"]]},"template":{"data":
{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
```

```
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
```

```
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Correlation Matrix for the Prediction of Disease
Progression"},"xaxis":{"anchor":"y","domain":[0,1]},"yaxis":
{"anchor":"x","autorange":"reversed","domain":[0,1]}}}
```


From the above matrix it can be found that the following attributes contribute more to the prediction of the disease progression.

bmi - positive correlation

bp - positive correlation

s3 - negative correlation

s4 - positive correlation

s5 - positive correlation

Data Preprocessing

```
# Splitting data into features and target variables.
X = diabetes_df_final[['bmi', 'bp', 's3', 's4', 's5']]
```

```
#X = diabetes_df_final.drop(columns=['target'])
y = diabetes_df_final['target']

# Initializing the Linear Regression model
model = LinearRegression()

# Initialize k-fold cross-validation
kf = KFold(n_splits=3, shuffle=True, random_state=42)
```

The reason for using cross validation is to summerise the all the results obtained when using different blocks for training and testing at different point of time.

```
# Initialize lists to store metrics
rmse_scores = []
r2_scores = []

# Perform k-fold cross-validation
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Calculate evaluation metrics
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    # Append scores to lists
    rmse_scores.append(rmse)
    r2_scores.append(r2)

# Calculating the average scores is becasuse of the cross validation
methodology
avg_rmse = np.mean(rmse_scores)
avg_r2 = np.mean(r2_scores)

# Printing the average scores from basic feature selection using
correlation matrix
print("Average RMSE from basic feature selection :", avg_rmse)
print("Average R^2 from basic feature selection:", avg_r2)
```

```
Average RMSE from basic feature selection : 56.57119967725297
Average R^2 from basic feature selection: 0.4562363008843464
```

Insights:

RMSE- Root Mean Squared Error metric gives the average magnitude of the error. In the above scenario, the error is approximately 55.4 units away/deviated from the actual values. Since the value is a bit on the higher side, model is less accurate in its predictions.

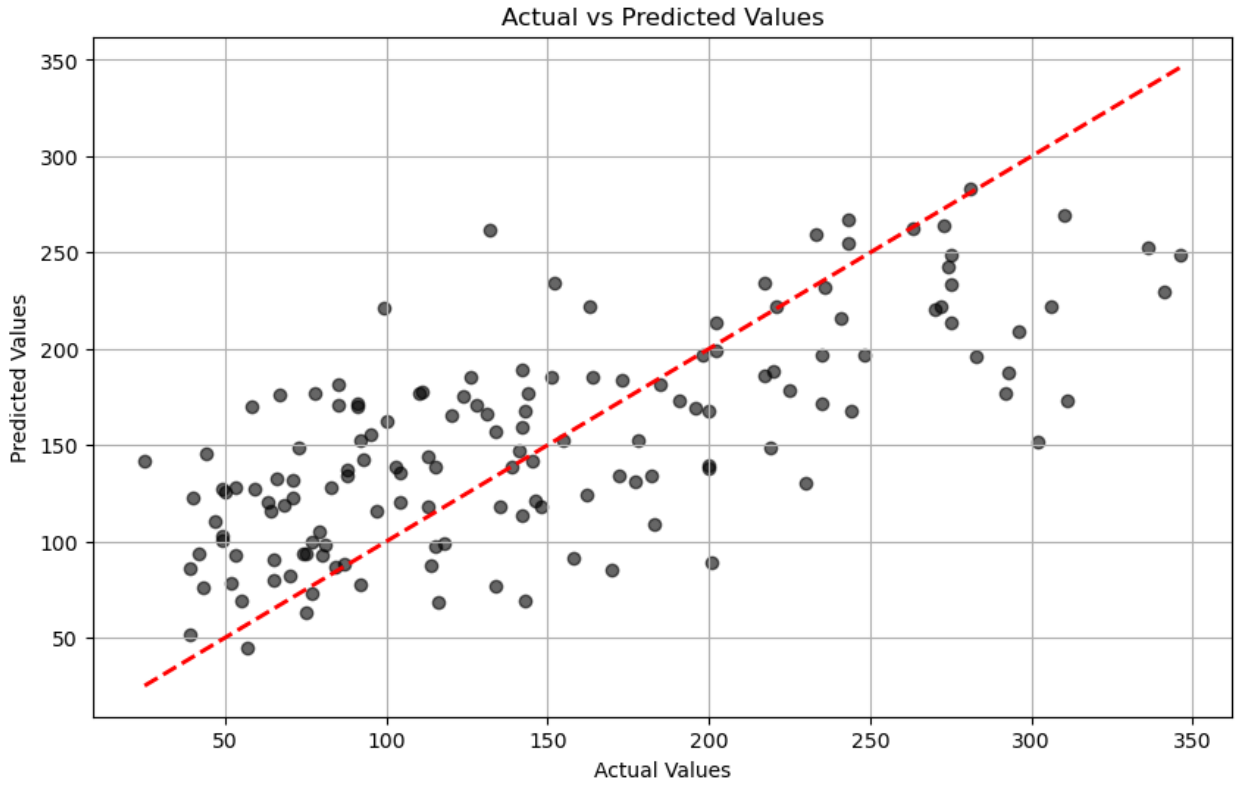
R^2 error value of 0.479 indicates that the model explains about only half(47.9%) of the variability of data in the model, indicating a moderate fit.

Overall, the model should be improved a little more such that the higher prediction capabilities are brought in.

Visualisations

Actual Vs Predicted Values

```
# Assuming y_test is the actual values and y_pred is the predicted
values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color="black", edgecolor="k", alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', lw=2)  # Line representing perfect prediction
plt.title("Actual vs Predicted Values")
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.show()
```



The graph displays the actual values plotted against the predicted values.Here the red dashed line represents the perfect prediction, where the predicted values would equal the actual values.

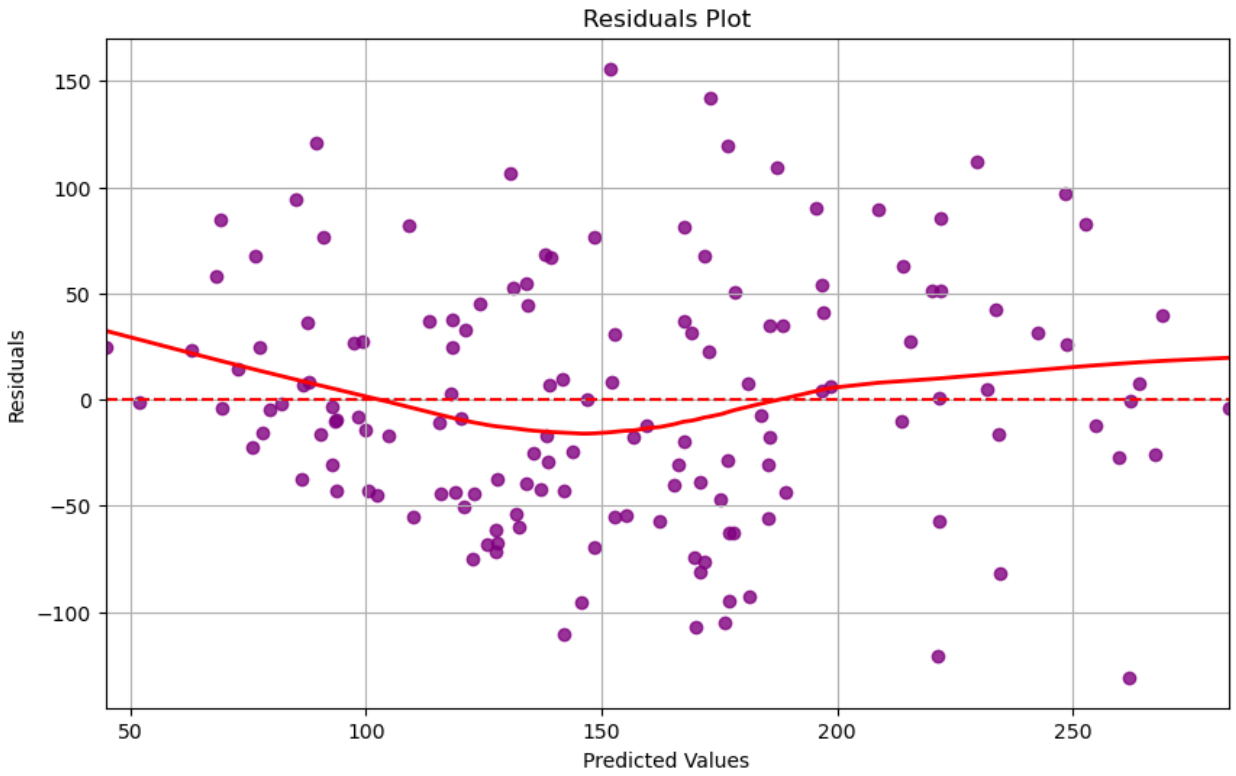
Insights:

There's a general upward trend, indicating that the model captures few of the relationships between features and target values.

However, there are many points deviating from the red line, indicating that the model isn't perfect and struggles with high variability in predictions. For example, for actual values between 50 and 150, the model predictions vary quite a bit, showing that the model has room for improvement in predictive accuracy.

Residual Plot

```
# Residuals = Actual values - Predicted values
residuals = y_test - y_pred

plt.figure(figsize=(10, 6))
sns.residplot(x=y_pred, y=residuals, lowess=True, color="purple",
line_kws={'color': 'red', 'lw': 2})
plt.title("Residuals Plot")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.grid(True)
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```



The residuals plot helps to evaluate the performance of the model by plotting residuals (differences between actual and predicted values) against predicted values.

Insights:

The residuals appear randomly distributed, but there is some pattern to them, as shown by the red fitted line. This slight curve suggests a potential issue with model fit (e.g., it could be a sign of non-linearity or that the model might not be fully capturing the underlying data structure).

The residuals are relatively close to zero in the middle range of predictions, but more variability is present in the lower and upper predicted values.

There is some systematic underestimation and overestimation, as the residuals are not centered uniformly around zero for all prediction ranges.

Summary

The model does a decent job overall but has areas where predictions deviate significantly, especially for extreme values.

The residuals plot suggests the model might not be accounting for some non-linear relationships, meaning that a more complex model or additional features may improve performance.

Model training using recursive feature elimination (RFE) method

```
X = diabetes_df_final.drop('target', axis=1)
y = diabetes_df_final['target']

#the model used as an estimator in RFE
model = LinearRegression()

rfe = RFE(estimator=model, n_features_to_select=5)
rfe = rfe.fit(X, y)

# Get the top 5 features
selected_features = X.columns[rfe.support_]
print(f"Selected Features: {selected_features}")
```

```
Selected Features: Index(['bmi', 'bp', 's1', 's2', 's5'],
dtype='object')
```

When implementing the RFE method, it is found the feature elimination is done in a loop until the number of rows to be selected is satisfied. During every iteration, the feature with the least coeffecient value is eliminated.

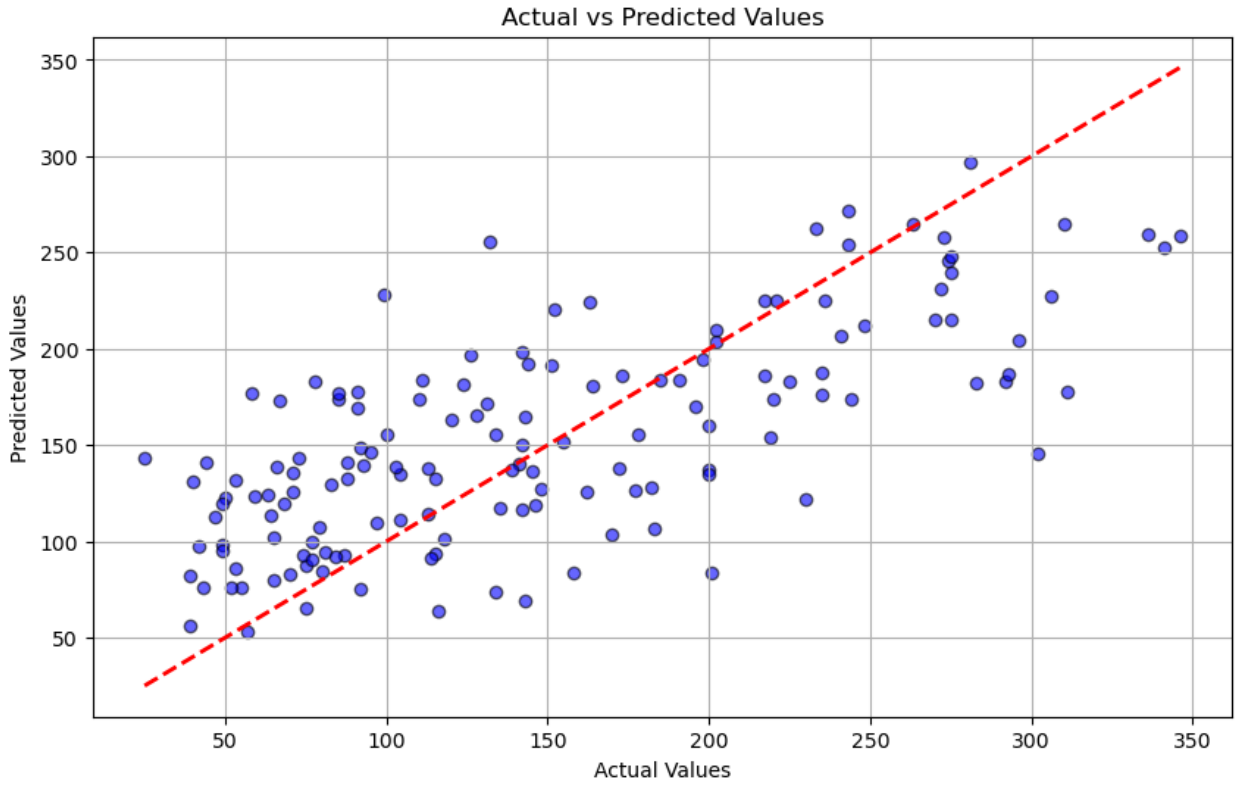
The 5 most contributing features are as follows: ['bmi', 'bp', 's1', 's2', 's5']

```
# Splitting data into features and target variables.
#X = diabetes_df_final[['bmi', 'bp', 's3', 's4', 's5']]
X = diabetes_df_final[selected_features]
y = diabetes_df_final['target']

# Initializing the Linear Regression model
model = LinearRegression()

# Initialize k-fold cross-validation
kf = KFold(n_splits=3, shuffle=True, random_state=42)

# Initialize lists to store metrics
rmse_scores_rfe = []
r2_scores_rfe = []

for train_indices, test_indices in kf.split(X):
    X_train_split, X_test_split = X.iloc[train_indices],
X.iloc[test_indices]
    y_train_split, y_test_split = y.iloc[train_indices],
y.iloc[test_indices]

    # Train the model
    model.fit(X_train_split, y_train_split)

    # Make predictions
    y_pred_split = model.predict(X_test_split)
```

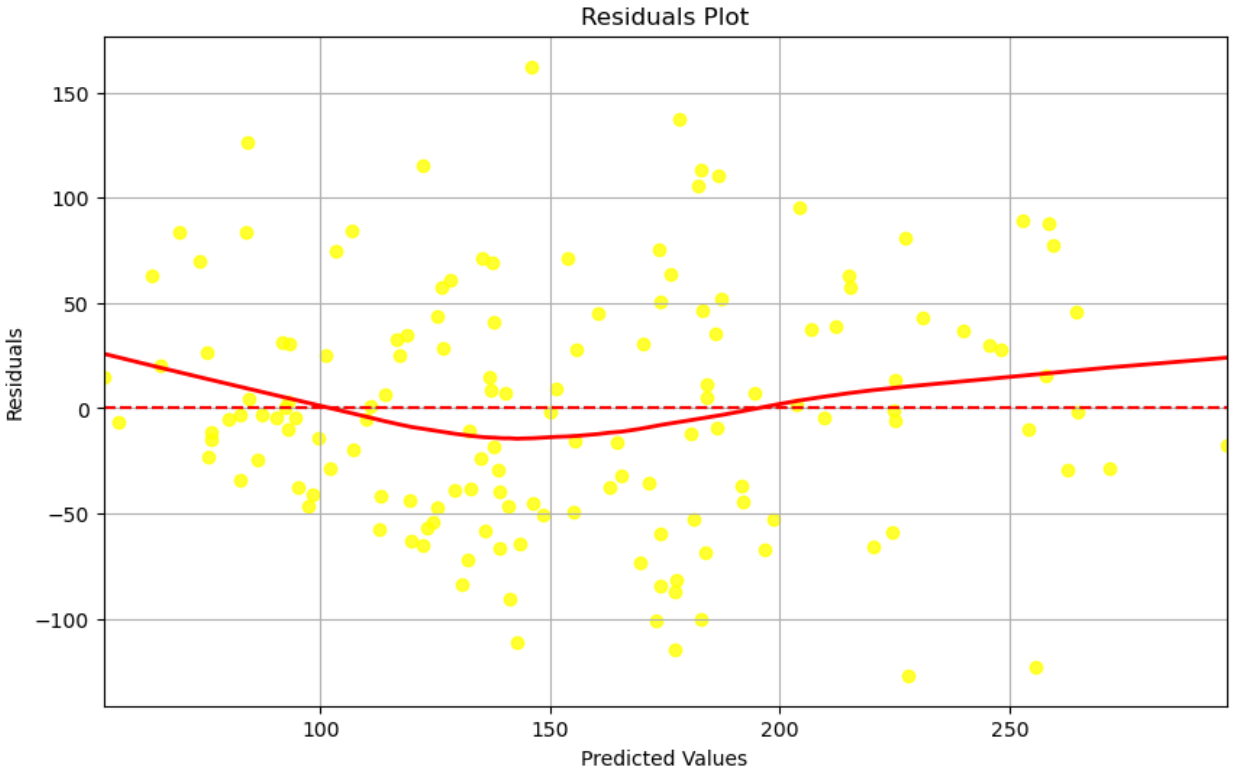
```
    # Calculate evaluation metrics
    rmse = np.sqrt(mean_squared_error(y_test_split, y_pred_split))
    r2 = r2_score(y_test_split, y_pred_split)

    # Append scores to lists
    rmse_scores_rfe.append(rmse)
    r2_scores_rfe.append(r2)

# Calculating the average scores
avg_rmse_rfe = np.mean(rmse_scores_rfe)
avg_r2_rfe = np.mean(r2_scores_rfe)

# Printing the average scores from RFE feature selection
print("Average RMSE from RFE feature selection :", avg_rmse_rfe)
print("Average R^2 from RFE feature selection:", avg_r2_rfe)
```

```
Average RMSE from RFE feature selection : 55.997681746628814
Average R^2 from RFE feature selection: 0.467805267975215
```

```
# Assuming y_test is the actual values and y_pred is the predicted
values
plt.figure(figsize=(10, 6))
plt.scatter(y_test_split, y_pred_split, color="blue", edgecolor="k",
alpha=0.6)
plt.plot([y_test_split.min(), y_test_split.max()],
[y_test_split.min(), y_test_split.max()], 'r--', lw=2)  # Line
representing perfect prediction
plt.title("Actual vs Predicted Values")
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.grid(True)
plt.show()
```




```
# Residuals = Actual values - Predicted values
residuals_rfe = y_test_split - y_pred_split

plt.figure(figsize=(10, 6))
sns.residplot(x=y_pred_split, y=residuals_rfe, lowess=True,
color="yellow", line_kws={'color': 'red', 'lw': 2})
plt.title("Residuals Plot")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.grid(True)
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```



Overall Picture:

The graph displaying the actual values plotted against the predicted values. the red dashed line represents the perfect prediction, where the predicted values would equal the actual values. The residuals are relatively close to zero in the middle range of predictions even when using RFE, but more variability is present in the lower and upper predicted values.

Therefore, the model predictions vary quite a bit and might not be fully capturing the underlying data structure, showing that the model has room for improvement in predictive accuracy.