Machine Learning Assignment No.4

Que.1 Using sklearn.datasets.load_diabetes apply Variance method for removing the constant column also after applying the Variance method apply multi linear regression on that data

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.feature_selection import VarianceThreshold
%matplotlib inline

from sklearn.datasets import load_diabetes

diabetes = load_diabetes()

diabetes

{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -
0.00259226,
          0.01990842, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286377, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
          0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986,  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 `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)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
```

```
  's6'],
 'data_filename': 'diabetes_data.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

```
diabetes.keys() # Keys which we can use
```

```
dict_keys(['data', 'target', 'frame', 'DESCR', 'feature_names',
'data_filename', 'target_filename', 'data_module'])
```

```
diabetes.data # Independent Data
```

```
array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]])
```

```
diabetes.target # Dependent Data
```

```
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.])
```

```
diabetes.feature_names # independent column names
```

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

```
diabetes.DESCR # Independent Column
```

```
'.. _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 `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)'
```

```
print(diabetes["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 `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)

diabetes


```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -
0.00259226,
          0.01990842, -0.01764613],
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286377, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
```

```
          0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986,  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 `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)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

```
#pd.DataFrame(Data,ColumnName)
df = pd.DataFrame(diabetes['data'],columns=diabetes['feature_names'])

df
```

```
          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
```

```
0.173816

           s4        s5        s6
0   -0.002592  0.019908 -0.017646
1   -0.039493 -0.068330 -0.092204
2   -0.002592  0.002864 -0.025930
3    0.034309  0.022692 -0.009362
4   -0.002592 -0.031991 -0.046641
..        ...       ...       ...
437 -0.002592  0.031193  0.007207
438  0.034309 -0.018118  0.044485
439 -0.011080 -0.046879  0.015491
440  0.026560  0.044528 -0.025930
441 -0.039493 -0.004220  0.003064

[442 rows x 10 columns]
```

```
diabetes["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.])
```

```python
df["target_values"] = diabetes.target # Dependent colummn CONTENT
```

```python
df
```

```
          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6  target_values
0   -0.002592  0.019908 -0.017646          151.0
1   -0.039493 -0.068330 -0.092204           75.0
2   -0.002592  0.002864 -0.025930          141.0
3    0.034309  0.022692 -0.009362          206.0
4   -0.002592 -0.031991 -0.046641          135.0
..        ...       ...       ...            ...
437 -0.002592  0.031193  0.007207          178.0
438  0.034309 -0.018118  0.044485          104.0
439 -0.011080 -0.046879  0.015491          132.0
440  0.026560  0.044528 -0.025930          220.0
441 -0.039493 -0.004220  0.003064           57.0

[442 rows x 11 columns]
```

```python
df.head()
```

```
        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 -0.005671 -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

         s4        s5        s6  target_values
0 -0.002592  0.019908 -0.017646          151.0
1 -0.039493 -0.068330 -0.092204           75.0
2 -0.002592  0.002864 -0.025930          141.0
3  0.034309  0.022692 -0.009362          206.0
4 -0.002592 -0.031991 -0.046641          135.0
```

```
df.tail()
```

```
          age       sex       bmi        bp        s1        s2
s3  \
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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6  target_values
437 -0.002592  0.031193  0.007207          178.0
438  0.034309 -0.018118  0.044485          104.0
439 -0.011080 -0.046879  0.015491          132.0
440  0.026560  0.044528 -0.025930          220.0
441 -0.039493 -0.004220  0.003064           57.0
```

```
df.shape
```

```
(442, 11)
```

```
df.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_values  442 non-null    float64
dtypes: float64(11)
memory usage: 38.1 KB
```

```
df.isnull().sum()
```

```
age              0
sex              0
bmi              0
bp               0
s1               0
s2               0
s3               0
s4               0
s5               0
s6               0
target_values    0
dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(df)
```

```
StandardScaler()
```

```
scaled_data = scaler.transform(df)
```

```
scaled_data
```

```
array([[ 0.80050009,  1.06548848,  1.29708846, ...,  0.41855058,
        -0.37098854, -0.01471948],
       [-0.03956713, -0.93853666, -1.08218016, ..., -1.43655059,
        -1.93847913, -1.00165882],
       [ 1.79330681,  1.06548848,  0.93453324, ...,  0.06020733,
        -0.54515416, -0.14457991],
       ...,
       [ 0.87686984,  1.06548848, -0.33441002, ..., -0.98558469,
         0.32567395, -0.26145431],
       [-0.9560041 , -0.93853666,  0.82123474, ...,  0.93615545,
        -0.54515416,  0.88131756],
       [-0.9560041 , -0.93853666, -1.53537419, ..., -0.08871747,
         0.06442552, -1.23540761]])
```

```
scaled_data.shape
```

```
(442, 11)
```

```
var_thres=VarianceThreshold(threshold=0.5)
var_thres.fit(scaled_data)
```

```
VarianceThreshold(threshold=0.5)
```

```
#which column is having good variaty of data means good variance
var_thres.get_support()
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

```
df.columns[var_thres.get_support() == True]
```

```
Index(['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6',
       'target_values'],
      dtype='object')
```

```
columns_having_var_more_than_50 = df.columns[var_thres.get_support()
== True]
```

```
columns_having_var_more_than_50
```

```
Index(['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6',
       'target_values'],
      dtype='object')
```

```
len(columns_having_var_more_than_50)
```

```
11
```

```
len(df.columns)
```

```
11
```

```
df.columns[var_thres.get_support() == False]
```

```
Index([], dtype='object')
```

```
columns_having_var_less_than_50 = df.columns[var_thres.get_support()
== False]
```

```
columns_having_var_less_than_50
```

```
Index([], dtype='object')
```

```
len(columns_having_var_less_than_50)
```

```
0
```

```
df.drop(target_values,inplace = True,axis= 1)
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Input In [40], in <cell line: 1>()
----> 1 df.drop(target_values,inplace = True,axis= 1)

NameError: name 'target_values' is not defined
```

Using the Model of Multi Linear Regression

```
cdf = df[['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5',
's6']] # Independent Variable

cdf

          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6
0   -0.002592  0.019908 -0.017646
1   -0.039493 -0.068330 -0.092204
2   -0.002592  0.002864 -0.025930
3    0.034309  0.022692 -0.009362
4   -0.002592 -0.031991 -0.046641
..        ...       ...       ...
437 -0.002592  0.031193  0.007207
438  0.034309 -0.018118  0.044485
```

```
439 -0.011080 -0.046879  0.015491
440  0.026560  0.044528 -0.025930
441 -0.039493 -0.004220  0.003064

[442 rows x 10 columns]
```

```python
cdf.head()
```

```
        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 -0.005671 -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


         s4        s5        s6
0 -0.002592  0.019908 -0.017646
1 -0.039493 -0.068330 -0.092204
2 -0.002592  0.002864 -0.025930
3  0.034309  0.022692 -0.009362
4 -0.002592 -0.031991 -0.046641
```

```python
cdf = df[["target_values"]] # Dependent variable
```

```python
cdf
```

```
     target_values
0            151.0
1             75.0
2            141.0
3            206.0
4            135.0
..             ...
437          178.0
438          104.0
439          132.0
440          220.0
441           57.0

[442 rows x 1 columns]
```

```python
X = df[['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5',
's6']] # Independent Variable
```

```python
X
```

```
          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6
0   -0.002592  0.019908 -0.017646
1   -0.039493 -0.068330 -0.092204
2   -0.002592  0.002864 -0.025930
3    0.034309  0.022692 -0.009362
4   -0.002592 -0.031991 -0.046641
..        ...       ...       ...
437 -0.002592  0.031193  0.007207
438  0.034309 -0.018118  0.044485
439 -0.011080 -0.046879  0.015491
440  0.026560  0.044528 -0.025930
441 -0.039493 -0.004220  0.003064

[442 rows x 10 columns]
```

```
Y = df[["target_values"]]
```

```
Y
```

```
     target_values
0            151.0
1             75.0
2            141.0
3            206.0
4            135.0
..             ...
```

```
437          178.0
438          104.0
439          132.0
440          220.0
441           57.0

[442 rows x 1 columns]
```

```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.2, random_state = 0)
```

```
X_train.shape
```

```
(353, 10)
```

```
X_test.shape
```

```
(89, 10)
```

```
Y_train.shape
```

```
(353, 1)
```

```
Y_test.shape
```

```
(89, 1)
```

```
# Feature Scaling

from sklearn.preprocessing import StandardScaler
sc_Y = StandardScaler()
Y_train = sc_Y.fit_transform(Y_train)
Y_test = sc_Y.transform(Y_test)
```

```
Y_train
```

```
array([[-0.85066765],
       [-0.18654485],
       [-1.25935861],
       [-1.2849018 ],
       [ 0.5797507 ],
       [-0.21208803],
       [-1.01669835],
       [ 0.72023822],
       [ 1.0906144 ],
       [-0.08437211],
       [-0.53137784],
       [ 1.79305198],
       [-0.69740854],
```

```
[ 0.5797507 ],
[ 0.0816586 ],
[ 0.14551656],
[-1.13164269],
[ 0.51589274],
[ 1.03952803],
[-1.06778472],
[ 0.10720178],
[ 0.33709044],
[-1.06778472],
[-0.54414943],
[-0.73572332],
[-0.4292051 ],
[ 1.57593491],
[ 1.57593491],
[-0.31426077],
[ 1.65256447],
[-1.02946995],
[-1.39984613],
[-1.02946995],
[-0.30148918],
[ 1.53762013],
[-0.21208803],
[-1.2849018 ],
[ 0.87349733],
[ 0.19660293],
[-0.9400688 ],
[-0.2631744 ],
[ 0.77132459],
[ 0.38817681],
[ 0.28600407],
[ 1.37158943],
[-0.95284039],
[-0.7740381 ],
[ 2.02294065],
[-0.86343925],
[-0.22485962],
[-0.63355058],
[-0.30148918],
[ 1.97185428],
[ 1.19278714],
[ 0.63083707],
[ 0.40094841],
[-0.51860625],
[-0.71018014],
[-0.33980396],
[ 0.38817681],
[ 0.32431885],
[-1.32321657],
[-0.69740854],
```

```
[ 1.37158943],
[ 1.74196561],
[ 1.15447236],
[-1.05501313],
[-1.54033364],
[ 0.03057223],
[-0.02051414],
[-1.2721302 ],
[ 1.39713262],
[-0.4292051 ],
[-0.63355058],
[-0.16100166],
[-1.2849018 ],
[-1.19550065],
[-0.0971437 ],
[ 0.33709044],
[-0.69740854],
[-0.9400688 ],
[-0.28871759],
[ 1.35881784],
[-0.35257555],
[ 0.59252229],
[ 0.42649159],
[-1.09332791],
[ 1.0906144 ],
[ 0.33709044],
[ 1.57593491],
[ 1.48653376],
[ 1.15447236],
[ 0.61806548],
[ 0.79686777],
[ 1.20555873],
[ 1.07784281],
[-0.85066765],
[-0.48029147],
[-0.74849491],
[-0.67186536],
[-1.01669835],
[ 1.51207695],
[-0.51860625],
[-0.87621084],
[-0.82512447],
[-1.39984613],
[ 0.26046089],
[-1.1060995 ],
[ 1.37158943],
[ 1.62702128],
[-0.13545848],
[-0.0971437 ],
[ 0.87349733],
```

```
[-0.78680969],
[-0.64632217],
[-1.25935861],
[-1.08055632],
[-1.01669835],
[-0.39089033],
[ 1.9335395 ],
[ 1.02675644],
[-0.14823007],
[ 0.49034955],
[ 0.88626892],
[-1.02946995],
[-0.45474829],
[-0.51860625],
[ 1.63979287],
[ 1.03952803],
[-0.9400688 ],
[-0.02051414],
[ 1.67810765],
[-1.11887109],
[-0.14823007],
[-1.1060995 ],
[ 0.93735529],
[ 0.69469503],
[-1.13164269],
[ 1.84413835],
[ 0.27323248],
[-0.85066765],
[-0.13545848],
[-1.29767339],
[-1.6169632 ],
[ 0.01780063],
[-1.23381543],
[-0.16100166],
[ 2.35500205],
[-1.00392676],
[-0.72295173],
[-0.54414943],
[-1.37430294],
[ 0.36263363],
[ 1.4226758 ],
[-0.04605733],
[-0.92729721],
[-1.1060995 ],
[-0.63355058],
[ 0.87349733],
[ 1.6014781 ],
[ 1.20555873],
[ 0.61806548],
[ 1.40990421],
```

```
[ 0.50312115],
[-0.69740854],
[ 0.41372   ],
[-0.85066765],
[ 1.23110192],
[-0.02051414],
[ 1.48653376],
[-1.18272906],
[-1.04224154],
[-0.81235288],
[-0.65909377],
[ 0.49034955],
[-0.49306306],
[-1.09332791],
[ 1.16724395],
[ 0.42649159],
[ 1.40990421],
[-1.32321657],
[ 0.10720178],
[ 0.83518255],
[ 0.7457814 ],
[-0.25040281],
[ 1.34604625],
[-0.6080074 ],
[-0.32703236],
[ 1.79305198],
[ 0.18383133],
[-0.87621084],
[-0.90175402],
[-0.0971437 ],
[ 1.65256447],
[-1.01669835],
[-1.43816091],
[-0.54414943],
[-1.16995746],
[ 1.35881784],
[ 0.33709044],
[ 0.20937452],
[-0.82512447],
[-0.95284039],
[ 0.82241096],
[ 0.69469503],
[-0.12268688],
[ 0.11997337],
[ 1.44821899],
[-1.16995746],
[ 0.61806548],
[ 1.44821899],
[ 1.53762013],
[-0.07160051],
```

```
[-0.73572332],
[-1.23381543],
[-1.05501313],
[-0.17377325],
[ 1.35881784],
[-0.10991529],
[ 0.26046089],
[-0.79958128],
[-1.05501313],
[ 0.60529389],
[-1.23381543],
[-1.36153135],
[ 1.44821899],
[-0.7740381 ],
[ 0.2349177 ],
[-1.23381543],
[ 0.64360866],
[ 0.04334382],
[-0.95284039],
[-0.95284039],
[-1.02946995],
[-0.36534714],
[-0.86343925],
[ 1.28218828],
[-1.2721302 ],
[-1.42538931],
[ 1.56316332],
[-0.10991529],
[ 1.19278714],
[-0.76126651],
[-0.00774255],
[-1.43816091],
[ 1.06507121],
[-0.76126651],
[ 1.29495988],
[-0.73572332],
[-0.90175402],
[ 2.48271797],
[-0.78680969],
[ 0.37540522],
[ 0.13274497],
[ 1.6014781 ],
[ 0.00502904],
[ 0.33709044],
[-0.35257555],
[-0.97838358],
[ 1.4226758 ],
[ 0.64360866],
[ 0.61806548],
[-0.55692103],
```

```
[-0.71018014],
[-1.16995746],
[-1.01669835],
[-0.56969262],
[-1.24658702],
[ 0.0816586 ],
[ 0.00502904],
[ 0.87349733],
[ 1.99739746],
[ 1.24387351],
[ 0.89904051],
[-1.1060995 ],
[ 0.27323248],
[-0.81235288],
[-1.01669835],
[ 0.15828815],
[-1.2721302 ],
[-0.46751988],
[ 0.61806548],
[-0.78680969],
[ 1.23110192],
[-1.46370409],
[ 1.00121325],
[-1.13164269],
[ 1.55039173],
[-1.15718587],
[-1.25935861],
[ 0.47757796],
[ 1.14170077],
[-0.4292051 ],
[ 1.28218828],
[-0.6080074 ],
[ 0.86072573],
[-0.46751988],
[ 2.30391568],
[-0.2631744 ],
[ 0.42649159],
[-1.13164269],
[-0.2631744 ],
[-0.81235288],
[ 0.45203478],
[ 0.56697911],
[-1.18272906],
[ 2.41886001],
[-0.54414943],
[-0.64632217],
[-0.49306306],
[-0.91452562],
[ 1.15447236],
[ 0.20937452],
```

```
       [-0.30148918],
       [ 1.03952803],
       [ 0.73300981],
       [ 0.93735529],
       [-0.87621084],
       [ 0.79686777],
       [-0.71018014],
       [-0.28871759],
       [-1.33598817],
       [ 0.98844166],
       [ 1.80582358],
       [-0.99115517],
       [ 0.64360866],
       [ 0.15828815],
       [ 0.64360866],
       [-1.18272906],
       [-0.7740381 ],
       [-0.40366192],
       [-0.00774255],
       [ 2.02294065],
       [-0.78680969],
       [-0.45474829],
       [-0.05882892],
       [-1.38707454],
       [-1.39984613],
       [-1.32321657],
       [-0.22485962],
       [-0.86343925],
       [-1.02946995],
       [-1.11887109],
       [-1.04224154],
       [ 2.02294065],
       [ 2.03571224],
       [-0.37811873],
       [ 1.16724395],
       [ 1.23110192],
       [-0.7740381 ],
       [ 1.65256447],
       [-0.12268688],
       [ 1.83136676]])
```

```
Y_test
```

```
array([[ 2.16342816],
       [ 0.80963936],
       [-0.31426077],
       [-1.11887109],
       [ 0.29877567],
       [ 1.57593491],
       [ 0.34986204],
       [ 1.02675644],
```

```
[-0.12268688],
[-0.67186536],
[ 1.28218828],
[ 0.28600407],
[-0.28871759],
[-0.99115517],
[ 1.43544739],
[-1.31044498],
[-0.83789606],
[-0.97838358],
[-0.64632217],
[ 0.04334382],
[ 0.2349177 ],
[ 1.5887065 ],
[-0.53137784],
[-0.19931644],
[-1.06778472],
[-0.30148918],
[-0.62077899],
[-0.74849491],
[ 0.50312115],
[ 0.56697911],
[ 0.83518255],
[ 0.37540522],
[ 0.20937452],
[ 0.61806548],
[ 0.86072573],
[ 1.65256447],
[-0.00774255],
[ 1.34604625],
[-1.31044498],
[ 0.59252229],
[-0.71018014],
[ 0.34986204],
[-0.72295173],
[ 0.59252229],
[ 1.18001555],
[-0.79958128],
[ 0.79686777],
[ 0.38817681],
[-0.86343925],
[ 1.51207695],
[ 0.05611541],
[-0.17377325],
[-0.49306306],
[-0.2631744 ],
[ 0.55420752],
[ 0.2476893 ],
[-0.37811873],
[-1.15718587],
```

```
       [ 1.00121325],
       [ 1.06507121],
       [-1.2721302 ],
       [-0.39089033],
       [-0.0971437 ],
       [-0.56969262],
       [-0.25040281],
       [ 1.92076791],
       [-1.25935861],
       [ 2.11234179],
       [-0.18654485],
       [-1.20827224],
       [-0.68463695],
       [ 0.2349177 ],
       [-0.81235288],
       [-0.78680969],
       [-1.08055632],
       [ 0.14551656],
       [-0.6080074 ],
       [ 0.43926318],
       [ 0.36263363],
       [ 1.67810765],
       [-0.13545848],
       [-0.02051414],
       [-1.33598817],
       [ 1.85690994],
       [-0.6080074 ],
       [-1.31044498],
       [-0.62077899],
       [-0.12268688],
       [-1.18272906]])
```

Multiple Regression Model

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X_train, Y_train)#training func question + answers
# The coefficients
print ('Intercept: ',regr.intercept_)
print ('Coefficient : ',regr.coef_)
```

```
Intercept:  [0.01190186]
Coefficient :  [[-0.45411743 -3.10565826  7.18726533  3.90136427 -8.46485465  4.14151165
   0.31650475  2.17539256  9.34469459  0.54954206]]
```

```
regr.intercept_
```

```
array([0.01190186])
```

```
regr.coef_
```

```
array([[-0.45411743, -3.10565826,  7.18726533,  3.90136427, -
8.46485465,
         4.14151165,  0.31650475,  2.17539256,  9.34469459,
0.54954206]])
```

```
regr.coef_[0][0]
```

```
-0.45411742742487105
```

```
regr.coef_[0][3]
```

```
3.901364270938186
```

```
y_pred = regr.predict(X_test)
```

```
y_pred
```

```
array([[ 1.10940719],
       [ 1.24300129],
       [ 0.15897835],
       [-0.39972899],
       [ 0.45745275],
       [ 1.37221077],
       [-0.48596763],
       [ 0.46577663],
       [-0.02694292],
       [ 1.07798329],
       [ 0.26360647],
       [ 0.34833887],
       [-0.54213766],
       [-0.75954119],
       [ 1.17146395],
       [-0.82056538],
       [ 0.05261681],
       [-1.08067458],
       [-0.65365173],
       [ 0.84915759],
       [ 0.57544513],
       [ 0.12378352],
       [ 0.12901306],
       [ 0.06282311],
       [ 0.59109143],
       [ 0.20400843],
       [-0.39414977],
       [-0.85272647],
       [ 0.51636895],
       [ 0.11495248],
       [ 0.30096962],
       [-0.86052308],
       [-0.07416053],
       [-0.07194107],
       [-0.13590689],
```

```
[ 0.57980448],
[ 0.18310623],
[ 0.49876686],
[-0.29861295],
[ 0.70005159],
[-0.85886048],
[ 0.15861575],
[-0.09579403],
[ 0.42245009],
[ 0.33456665],
[-0.98695903],
[-0.10524052],
[-0.16512355],
[-0.39329827],
[ 1.05667416],
[ 0.13202851],
[-0.98418766],
[ 0.03975565],
[ 0.06619027],
[ 1.09600747],
[ 0.28894832],
[ 0.50161571],
[-0.41664124],
[-0.24779498],
[ 0.21610193],
[ 0.8063501 ],
[ 0.25309987],
[ 0.07366484],
[-0.54581902],
[ 1.34685464],
[ 0.00729946],
[-0.88340297],
[ 1.02123222],
[ 0.655185  ],
[-1.33364583],
[-0.93407196],
[-0.28486428],
[-0.60031211],
[-0.08881658],
[-0.24683005],
[ 0.49087755],
[-0.69031506],
[ 0.58637801],
[ 0.86247663],
[ 0.44102528],
[-0.02550618],
[ 0.72565075],
[-1.3667631 ],
[ 0.69736754],
[-0.95572962],
```

```
       [-0.72371204],
       [-0.08059804],
       [ 0.54191823],
       [-0.24037272]])
```

```
from sklearn.metrics import r2_score
```

```
print(f"R2 Score : {r2_score(Y_test,y_pred)*100} % ")
```

```
R2 Score : 33.222203269065155 %
```

```
print(f"Mean absolute error: {np.mean(np.absolute(y_pred - Y_test))}
")#pred - actual
```

```
Mean absolute error: 0.589718094672523
```

```
print("Residual sum of squares (MSE): %.2f" % np.mean((y_pred -
Y_test) **2))
```

```
Residual sum of squares (MSE): 0.56
```

Que.2 Using sklearn.datasets.load_wine Apply Correlation and make a heat map using seaborn and remove the highly correlated columns if exist and the apply SVM and get the best accuracy by changing the Hyperparameters

In this step we will be removing the features which are highly correlated

```
#importing libraries
from sklearn.datasets import load_wine
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
data = load_wine()
```

```
data
```

```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00,
3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
```

```
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
        2, 2]),
 'frame': None,
 'target_names': array(['class_0', 'class_1', 'class_2'],
dtype='<U7'),
 'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\
n------------------------\n\n**Data Set Characteristics:**\n\
n    :Number of Instances: 178 (50 in each of three classes)\
n    :Number of Attributes: 13 numeric, predictive attributes and the
class\n    :Attribute Information:\n \t\t- Alcohol\n \t\t- Malic acid\
n \t\t- Ash\n\t\t- Alcalinity of ash  \n \t\t- Magnesium\n\t\t- Total
phenols\n \t\t- Flavanoids\n \t\t- Nonflavanoid phenols\n \t\t-
Proanthocyanins\n\t\t- Color intensity\n \t\t- Hue\n \t\t- OD280/OD315
of diluted wines\n \t\t- Proline\n\n    - class:\n            -
class_0\n            - class_1\n            - class_2\n\t\t\
n    :Summary Statistics:\n    \n    ============================
==== ===== ======= =====\n                                   Min   Max
Mean     SD\n    ============================ ==== ===== =======
=====\n    Alcohol:                          11.0  14.8    13.0   0.8\n
Malic Acid:                  0.74  5.80    2.34  1.12\n    Ash:
1.36  3.23    2.36  0.27\n    Alcalinity of Ash:            10.6  30.0
19.5   3.3\n    Magnesium:                    70.0 162.0    99.7
14.3\n    Total Phenols:                0.98  3.88    2.29  0.63\n
Flavanoids:                  0.34  5.08    2.03  1.00\n
Nonflavanoid Phenols:        0.13  0.66    0.36  0.12\n
Proanthocyanins:             0.41  3.58    1.59  0.57\n    Colour
Intensity:              1.3  13.0     5.1   2.3\n    Hue:
0.48  1.71    0.96  0.23\n    OD280/OD315 of diluted wines: 1.27  4.00
2.61  0.71\n    Proline:                       278  1680     746
315\n    ============================ ==== ===== ======= =====\n\n
:Missing Attribute Values: None\n    :Class Distribution: class_0
(59), class_1 (71), class_2 (48)\n    :Creator: R.A. Fisher\
```

```
n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\
n    :Date: July, 1988\n\nThis is a copy of UCI ML Wine recognition
datasets.\nhttps://archive.ics.uci.edu/ml/machine-learning-databases/
wine/wine.data\n\nThe data is the results of a chemical analysis of
wines grown in the same\nregion in Italy by three different
cultivators. There are thirteen different\nmeasurements taken for
different constituents found in the three types of\nwine.\n\nOriginal
Owners: \n\nForina, M. et al, PARVUS - \nAn Extendible Package for
Data Exploration, Classification and Correlation. \nInstitute of
Pharmaceutical and Food Analysis and Technologies,\nVia Brigata
Salerno, 16147 Genoa, Italy.\n\nCitation:\n\nLichman, M. (2013). UCI
Machine Learning Repository\n[https://archive.ics.uci.edu/ml]. Irvine,
CA: University of California,\nSchool of Information and Computer
Science. \n\n.. topic:: References\n\n  (1) S. Aeberhard, D. Coomans
and O. de Vel, \n  Comparison of Classifiers in High Dimensional
Settings, \n  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science
and Dept. of  \n  Mathematics and Statistics, James Cook University of
North Queensland. \n  (Also submitted to Technometrics). \n\n  The
data was used with many others for comparing various \n  classifiers.
The classes are separable, though only RDA \n  has achieved 100%
correct classification. \n  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN
96.1% (z-transformed data)) \n  (All results using the leave-one-out
technique) \n\n  (2) S. Aeberhard, D. Coomans and O. de Vel, \n  "THE
CLASSIFICATION PERFORMANCE OF RDA" \n  Tech. Rep. no. 92-01, (1992),
Dept. of Computer Science and Dept. of \n  Mathematics and Statistics,
James Cook University of North Queensland. \n  (Also submitted to
Journal of Chemometrics).\n',
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline']}
```

```
data.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR',
'feature_names'])
```

```
data.feature_names
```

```
['alcohol',
 'malic_acid',
 'ash',
```

```
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

```
data.data # Independent Variable
```

```
array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

```
data.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2])
```

```
columns_name = data.feature_names
```

```
columns_name
```

```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']

# Data === data.data (independent columns) , column_name ==
data.feature_names

df = pd.DataFrame(data.data, columns = columns_name)# IV CONTENT
ASWELL AS IV COLUMN NAMES

df

     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
```

```
1           2.76                      0.26             1.28
4.38  1.05
2           3.24                      0.30             2.81
5.68  1.03
3           3.49                      0.24             2.18
7.80  0.86
4           2.69                      0.39             1.82
4.32  1.04
..           ...                       ...              ...              ..
.   ...
173         0.61                      0.52             1.06
7.70  0.64
174         0.75                      0.43             1.41
7.30  0.70
175         0.69                      0.43             1.35
10.20  0.59
176         0.68                      0.53             1.46
9.30  0.60
177         0.76                      0.56             1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

```
data.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
2,
```

```
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2])
```

```
df["target_values"] = data.target #dependent colummn CONTENT

df # Dependent Variable as well as Independent Variable
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
1          2.76                  0.26             1.28
4.38  1.05
2          3.24                  0.30             2.81
5.68  1.03
3          3.49                  0.24             2.18
7.80  0.86
4          2.69                  0.39             1.82
4.32  1.04
..          ...                   ...              ...              ..
.   ...
173        0.61                  0.52             1.06
7.70  0.64
```

```
174        0.75                   0.43            1.41
7.30  0.70
175        0.69                   0.43            1.35
10.20  0.59
176        0.68                   0.53            1.46
9.30  0.60
177        0.76                   0.56            1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline  target_values
0                            3.92   1065.0              0
1                            3.40   1050.0              0
2                            3.17   1185.0              0
3                            3.45   1480.0              0
4                            2.93    735.0              0
..                            ...      ...            ...
173                          1.74    740.0              2
174                          1.56    750.0              2
175                          1.56    835.0              2
176                          1.62    840.0              2
177                          1.60    560.0              2

[178 rows x 14 columns]
```

```
df.head()
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0    14.23        1.71  2.43               15.6      127.0
2.80
1    13.20        1.78  2.14               11.2      100.0
2.65
2    13.16        2.36  2.67               18.6      101.0
2.80
3    14.37        1.95  2.50               16.8      113.0
3.85
4    13.24        2.59  2.87               21.0      118.0
2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity
hue  \
0        3.06                  0.28             2.29             5.64
1.04
1        2.76                  0.26             1.28             4.38
1.05
2        3.24                  0.30             2.81             5.68
1.03
3        3.49                  0.24             2.18             7.80
0.86
4        2.69                  0.39             1.82             4.32
```

```
1.04

   od280/od315_of_diluted_wines  proline  target_values
0                          3.92   1065.0              0
1                          3.40   1050.0              0
2                          3.17   1185.0              0
3                          3.45   1480.0              0
4                          2.93    735.0              0
```

df.tail()

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
173        0.61                  0.52             1.06
7.7  0.64
174        0.75                  0.43             1.41
7.3  0.70
175        0.69                  0.43             1.35
10.2  0.59
176        0.68                  0.53             1.46
9.3  0.60
177        0.76                  0.56             1.35
9.2  0.61

     od280/od315_of_diluted_wines  proline  target_values
173                          1.74    740.0              2
174                          1.56    750.0              2
175                          1.56    835.0              2
176                          1.62    840.0              2
177                          1.60    560.0              2
```

df.isnull().sum()

```
alcohol                        0
malic_acid                     0
ash                            0
alcalinity_of_ash              0
magnesium                      0
```

```
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
target_values                   0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target_values                 178 non-null    int32
dtypes: float64(13), int32(1)
memory usage: 18.9 KB
```

```
df.shape
```

```
(178, 14)
```

```
X = df.drop("target_values",axis=1)
Y = df["target_values"]
```

```
X
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
```

```
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
1          2.76                  0.26             1.28
4.38  1.05
2          3.24                  0.30             2.81
5.68  1.03
3          3.49                  0.24             2.18
7.80  0.86
4          2.69                  0.39             1.82
4.32  1.04
..          ...                   ...              ...              ..
.   ...
173        0.61                  0.52             1.06
7.70  0.64
174        0.75                  0.43             1.41
7.30  0.70
175        0.69                  0.43             1.35
10.20  0.59
176        0.68                  0.53             1.46
9.30  0.60
177        0.76                  0.56             1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
```

```
174                              1.56    750.0
175                              1.56    835.0
176                              1.62    840.0
177                              1.60    560.0

[178 rows x 13 columns]
```

```
Y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
173    2
174    2
175    2
176    2
177    2
Name: target_values, Length: 178, dtype: int32
```

```
Y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: target_values, dtype: int32
```

```
# separate dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
    X,
    Y,
    test_size=0.3,
    random_state=0)
```

```
X_train.shape, X_test.shape
```

```
((124, 13), (54, 13))
```

```
X_train
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
22     13.71        1.86  2.36               16.6      101.0
2.61
108    12.22        1.29  1.94               19.0       92.0
2.36
175    13.27        4.28  2.26               20.0      120.0
```

```
1.59
145    13.16        3.57  2.15               21.0      102.0
1.50
71     13.86        1.51  2.67               25.0       86.0
2.95
..       ...         ...   ...                ...        ...
...
103    11.82        1.72  1.88               19.5       86.0
2.50
67     12.37        1.17  1.92               19.6       78.0
2.11
117    12.42        1.61  2.19               22.5      108.0
2.00
47     13.90        1.68  2.12               16.0      101.0
3.10
172    14.16        2.51  2.48               20.0       91.0
1.68

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
22         2.88                  0.27             1.69
3.80  1.11
108        2.04                  0.39             2.08
2.70  0.86
175        0.69                  0.43             1.35
10.20  0.59
145        0.55                  0.43             1.30
4.00  0.60
71         2.86                  0.21             1.87
3.38  1.36
..          ...                   ...              ...                ..
.   ...
103        1.64                  0.37             1.42
2.06  0.94
67         2.00                  0.27             1.04
4.68  1.12
117        2.09                  0.34             1.61
2.06  1.06
47         3.39                  0.21             2.14
6.10  0.91
172        0.70                  0.44             1.24
9.70  0.62

     od280/od315_of_diluted_wines  proline
22                           4.00   1035.0
108                          3.02    312.0
175                          1.56    835.0
145                          1.68    830.0
71                           3.16    410.0
..                            ...      ...
```

```
103                           2.44    415.0
67                            3.48    510.0
117                           2.96    345.0
47                            3.33    985.0
172                           1.71    660.0

[124 rows x 13 columns]
```

```
X_train.shape
```

```
(124, 13)
```

```
X_train.corr()
```

```
                               alcohol  malic_acid       ash  \
alcohol                       1.000000    0.087268  0.228809
malic_acid                    0.087268    1.000000  0.200015
ash                           0.228809    0.200015  1.000000
alcalinity_of_ash            -0.326030    0.304109  0.446093
magnesium                     0.212436   -0.059823  0.181737
total_phenols                 0.352899   -0.298813  0.121369
flavanoids                    0.296712   -0.408887  0.060808
nonflavanoid_phenols         -0.167773    0.363213  0.185052
proanthocyanins               0.095713   -0.190354 -0.025868
color_intensity               0.565029    0.305012  0.243573
hue                          -0.047430   -0.545493 -0.108399
od280/od315_of_diluted_wines  0.073438   -0.390354 -0.018053
proline                       0.627676   -0.200906  0.158194

                              alcalinity_of_ash  magnesium
total_phenols  \
alcohol                               -0.326030   0.212436
0.352899
malic_acid                             0.304109  -0.059823      -
0.298813
ash                                    0.446093   0.181737
0.121369
alcalinity_of_ash                      1.000000  -0.088590      -
0.367199
magnesium                             -0.088590   1.000000
0.163801
total_phenols                         -0.367199   0.163801
1.000000
flavanoids                            -0.414673   0.143421
0.874093
nonflavanoid_phenols                   0.398878  -0.305155      -
0.450308
proanthocyanins                       -0.255579   0.270090
0.614683
color_intensity                       -0.030653   0.125051      -
0.068791
```

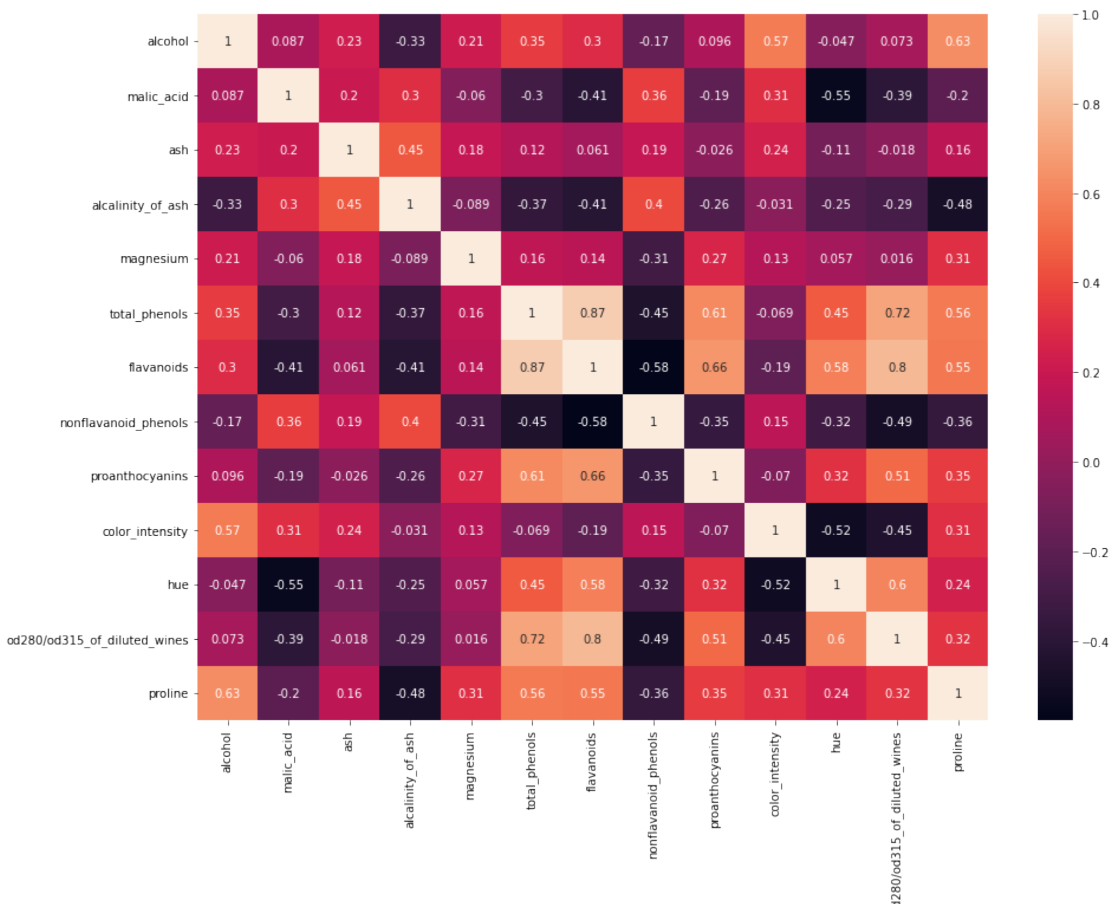
```
hue                                   -0.251091   0.057459
0.453501
od280/od315_of_diluted_wines          -0.287010   0.015833
0.716321
proline                               -0.481131   0.312073
0.558725

                              flavanoids  nonflavanoid_phenols  \
alcohol                         0.296712             -0.167773
malic_acid                     -0.408887              0.363213
ash                             0.060808              0.185052
alcalinity_of_ash              -0.414673              0.398878
magnesium                       0.143421             -0.305155
total_phenols                   0.874093             -0.450308
flavanoids                      1.000000             -0.578595
nonflavanoid_phenols           -0.578595              1.000000
proanthocyanins                 0.660619             -0.351086
color_intensity                -0.190290              0.153267
hue                             0.578615             -0.315259
od280/od315_of_diluted_wines    0.795590             -0.489811
proline                         0.553097             -0.361626

                              proanthocyanins  color_intensity
hue  \
alcohol                              0.095713         0.565029 -
0.047430
malic_acid                          -0.190354         0.305012 -
0.545493
ash                                 -0.025868         0.243573 -
0.108399
alcalinity_of_ash                   -0.255579        -0.030653 -
0.251091
magnesium                            0.270090         0.125051
0.057459
total_phenols                        0.614683        -0.068791
0.453501
flavanoids                           0.660619        -0.190290
0.578615
nonflavanoid_phenols                -0.351086         0.153267 -
0.315259
proanthocyanins                      1.000000        -0.069615
0.320218
color_intensity                     -0.069615         1.000000 -
0.519728
hue                                  0.320218        -0.519728
1.000000
od280/od315_of_diluted_wines         0.506250        -0.448015
0.595385
proline                              0.352654         0.313506
0.243301
```

```
                              od280/od315_of_diluted_wines   proline
alcohol                                           0.073438  0.627676
malic_acid                                       -0.390354 -0.200906
ash                                              -0.018053  0.158194
alcalinity_of_ash                                -0.287010 -0.481131
magnesium                                         0.015833  0.312073
total_phenols                                     0.716321  0.558725
flavanoids                                        0.795590  0.553097
nonflavanoid_phenols                             -0.489811 -0.361626
proanthocyanins                                   0.506250  0.352654
color_intensity                                  -0.448015  0.313506
hue                                               0.595385  0.243301
od280/od315_of_diluted_wines                      1.000000  0.321756
proline                                           0.321756  1.000000
```

```python
import seaborn as sns
# Using Pearson Correlation
plt.figure(figsize=(15,11))
cor = X_train.corr()
sns.heatmap(cor, annot=True)
plt.show()
```



Access the below diagonal elements

```
import numpy as np
arr =  np.array([[11,22,33],
                [44,55,66],
                [77,88,99]])

arr[0][1]

22

arr[0][0]

11

arr[1][0]

44

# Using for loop access the elements

for row in range(len(arr)):
    for col in range(len(arr)):
        print(f"{arr[row][col]}")

11
22
33
44
55
66
77
88
99

# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything
other feature

def correlation(dataset, threshold):# X_train,0.5
    col_corr = set()  # Set of all the names of correlated columns
    col_corr_lst = []
    print(f"set initial {col_corr}")
    print(f"list initial {col_corr_lst}")
    corr_arr = dataset.corr() # corr_arr is my correlaion matrix which
is 2d
    for row in range(len(corr_arr)):
        for col in range(row):
            if abs(corr_arr.iloc[row, col]) > threshold: # we are
interested in absolute coeff value
                colname = corr_arr.columns[row]  # getting the name of
column
                col_corr_lst.append(colname)
```

```
                col_corr.add(colname)
                print(f"colname name which is correlated is 
{colname}")
                print(f"set {col_corr}")
                print(f"lst {col_corr_lst}")


    print(f"list is {col_corr_lst}")
    return col_corr

corr_features = correlation(X_train, 0.5)#data,threshold
len(set(corr_features))
```

```
set initial set()
list initial []
colname name which is correlated is flavanoids
set {'flavanoids'}
lst ['flavanoids']
colname name which is correlated is nonflavanoid_phenols
set {'nonflavanoid_phenols', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols']
colname name which is correlated is proanthocyanins
set {'proanthocyanins', 'nonflavanoid_phenols', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins']
colname name which is correlated is proanthocyanins
set {'proanthocyanins', 'nonflavanoid_phenols', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 
'proanthocyanins']
colname name which is correlated is color_intensity
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 
'proanthocyanins', 'color_intensity']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 
'hue', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 
'proanthocyanins', 'color_intensity', 'hue']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 
'hue', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 
'proanthocyanins', 'color_intensity', 'hue', 'hue']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 
'hue', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue']
colname name which is correlated is od280/od315_of_diluted_wines
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 
```

```
'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines']
colname name which is correlated is od280/od315_of_diluted_wines
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols',
'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines']
colname name which is correlated is od280/od315_of_diluted_wines
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols',
'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines']
colname name which is correlated is od280/od315_of_diluted_wines
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols',
'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins',
'nonflavanoid_phenols', 'hue', 'od280/od315_of_diluted_wines',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'proline']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins',
'nonflavanoid_phenols', 'hue', 'od280/od315_of_diluted_wines',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'proline', 'proline']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins',
'nonflavanoid_phenols', 'hue', 'od280/od315_of_diluted_wines',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
```

```
'proline', 'proline', 'proline']
list is ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
'proanthocyanins', 'color_intensity', 'hue', 'hue', 'hue',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines',
'proline', 'proline', 'proline']

7
```

```
corr_features
```

```
{'color_intensity',
 'flavanoids',
 'hue',
 'nonflavanoid_phenols',
 'od280/od315_of_diluted_wines',
 'proanthocyanins',
 'proline'}
```

```
X_train.drop(corr_features,axis=1,inplace = True)
X_test.drop(corr_features,axis=1,inplace = True)
```

```
X_train
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols
22     13.71        1.86  2.36               16.6      101.0
2.61
108    12.22        1.29  1.94               19.0       92.0
2.36
175    13.27        4.28  2.26               20.0      120.0
1.59
145    13.16        3.57  2.15               21.0      102.0
1.50
71     13.86        1.51  2.67               25.0       86.0
2.95
..       ...         ...   ...                ...        ...
...
103    11.82        1.72  1.88               19.5       86.0
2.50
67     12.37        1.17  1.92               19.6       78.0
2.11
117    12.42        1.61  2.19               22.5      108.0
2.00
47     13.90        1.68  2.12               16.0      101.0
3.10
172    14.16        2.51  2.48               20.0       91.0
1.68

[124 rows x 6 columns]
```

```
X_train.head()
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  
total_phenols  
22     13.71        1.86  2.36               16.6      101.0  
2.61  
108    12.22        1.29  1.94               19.0       92.0  
2.36  
175    13.27        4.28  2.26               20.0      120.0  
1.59  
145    13.16        3.57  2.15               21.0      102.0  
1.50  
71     13.86        1.51  2.67               25.0       86.0  
2.95  
```

```
X_train.tail()
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  
total_phenols  
103    11.82        1.72  1.88               19.5       86.0  
2.50  
67     12.37        1.17  1.92               19.6       78.0  
2.11  
117    12.42        1.61  2.19               22.5      108.0  
2.00  
47     13.90        1.68  2.12               16.0      101.0  
3.10  
172    14.16        2.51  2.48               20.0       91.0  
1.68  
```

```
X_test
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  
total_phenols  
54     13.74        1.67  2.25               16.4      118.0  
2.60  
151    12.79        2.67  2.48               22.0      112.0  
1.48  
63     12.37        1.13  2.16               19.0       87.0  
3.50  
55     13.56        1.73  2.46               20.5      116.0  
2.96  
123    13.05        5.80  2.13               21.5       86.0  
2.62  
121    11.56        2.05  3.23               28.5      119.0  
3.18  
7      14.06        2.15  2.61               17.6      121.0  
2.60  
160    12.36        3.83  2.38               21.0       88.0  
2.30  
106    12.25        1.73  2.12               19.0       80.0  
1.65  
90     12.08        1.83  2.32               18.5       81.0  
```

1.60					
141	13.36	2.56	2.35	20.0	89.0
1.40					
146	13.88	5.04	2.23	20.0	80.0
0.98					
5	14.20	1.76	2.45	15.2	112.0
3.27					
98	12.37	1.07	2.10	18.5	88.0
3.52					
168	13.58	2.58	2.69	24.5	105.0
1.55					
80	12.00	0.92	2.00	19.0	86.0
2.42					
33	13.76	1.53	2.70	19.5	132.0
2.95					
18	14.19	1.59	2.48	16.5	108.0
3.30					
61	12.64	1.36	2.02	16.8	100.0
2.02					
51	13.83	1.65	2.60	17.2	94.0
2.45					
66	13.11	1.01	1.70	15.0	78.0
2.98					
37	13.05	1.65	2.55	18.0	98.0
2.45					
4	13.24	2.59	2.87	21.0	118.0
2.80					
104	12.51	1.73	1.98	20.5	85.0
2.20					
60	12.33	1.10	2.28	16.0	101.0
2.05					
111	12.52	2.43	2.17	21.0	88.0
2.55					
126	12.43	1.53	2.29	21.5	86.0
2.74					
86	12.16	1.61	2.31	22.8	90.0
1.78					
112	11.76	2.68	2.92	20.0	103.0
1.75					
164	13.78	2.76	2.30	22.0	90.0
1.35					
26	13.39	1.77	2.62	16.1	93.0
2.85					
56	14.22	1.70	2.30	16.3	118.0
3.20					
129	12.04	4.30	2.38	22.0	80.0
2.10					
45	14.21	4.04	2.44	18.9	111.0
2.85					
8	14.83	1.64	2.17	14.0	97.0

```
2.80
44     13.05        1.77  2.10                 17.0      107.0
3.00
161    13.69        3.26  2.54                 20.0      107.0
1.83
92     12.69        1.53  2.26                 20.7       80.0
1.38
94     11.62        1.99  2.28                 18.0       98.0
3.02
174    13.40        3.91  2.48                 23.0      102.0
1.80
24     13.50        1.81  2.61                 20.0       96.0
2.53
30     13.73        1.50  2.70                 22.5      101.0
3.00
93     12.29        2.83  2.22                 18.0       88.0
2.45
101    12.60        1.34  1.90                 18.5       88.0
1.45
113    11.41        0.74  2.50                 21.0       88.0
2.48
19     13.64        3.10  2.56                 15.2      116.0
2.70
135    12.60        2.46  2.20                 18.5       94.0
1.62
74     11.96        1.09  2.30                 21.0      101.0
3.38
144    12.25        3.88  2.20                 18.5      112.0
1.38
16     14.30        1.92  2.72                 20.0      120.0
2.80
131    12.88        2.99  2.40                 20.0      104.0
1.30
138    13.49        3.59  2.19                 19.5       88.0
1.62
40     13.56        1.71  2.31                 16.2      117.0
3.15
158    14.34        1.68  2.70                 25.0       98.0
2.80
```

```
X_test.head()
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols
54     13.74        1.67  2.25                 16.4      118.0
2.60
151    12.79        2.67  2.48                 22.0      112.0
1.48
63     12.37        1.13  2.16                 19.0       87.0
3.50
55     13.56        1.73  2.46                 20.5      116.0
```

```
2.96
123    13.05        5.80  2.13                 21.5       86.0
2.62
```

```
X_test.tail()
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols
16     14.30        1.92  2.72                 20.0      120.0
2.80
131    12.88        2.99  2.40                 20.0      104.0
1.30
138    13.49        3.59  2.19                 19.5       88.0
1.62
40     13.56        1.71  2.31                 16.2      117.0
3.15
158    14.34        1.68  2.70                 25.0       98.0
2.80
```

```
df =
df.drop(["flavanoids","nonflavanoid_phenols","proanthocyanins","color_
intensity","hue","od280/od315_of_diluted_wines","proline"],axis = 1)
df
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43                 15.6      127.0
2.80
1      13.20        1.78  2.14                 11.2      100.0
2.65
2      13.16        2.36  2.67                 18.6      101.0
2.80
3      14.37        1.95  2.50                 16.8      113.0
3.85
4      13.24        2.59  2.87                 21.0      118.0
2.80
..       ...         ...   ...                  ...        ...
...
173    13.71        5.65  2.45                 20.5       95.0
1.68
174    13.40        3.91  2.48                 23.0      102.0
1.80
175    13.27        4.28  2.26                 20.0      120.0
1.59
176    13.17        2.59  2.37                 20.0      120.0
1.65
177    14.13        4.10  2.74                 24.5       96.0
2.05

     target_values
0                0
```

```
1                0
2                0
3                0
4                0
..             ...
173              2
174              2
175              2
176              2
177              2

[178 rows x 7 columns]
```

Using the SVM Model

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

df

     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     target_values
0                0
```

```
1                0
2                0
3                0
4                0
..             ...
173              2
174              2
175              2
176              2
177              2

[178 rows x 7 columns]
```

df.head()

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0    14.23        1.71  2.43               15.6      127.0
2.80
1    13.20        1.78  2.14               11.2      100.0
2.65
2    13.16        2.36  2.67               18.6      101.0
2.80
3    14.37        1.95  2.50               16.8      113.0
3.85
4    13.24        2.59  2.87               21.0      118.0
2.80

   target_values
0              0
1              0
2              0
3              0
4              0
```

df.tail()

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     target_values
```

```
173             2
174             2
175             2
176             2
177             2
```

```
feature_names =
df[["alcohol","malic_acid","ash","alcalinity_of_ash","magnesium","tota
l_phenols"]] # Independent Variable
```

```
feature_names
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

[178 rows x 6 columns]
```

```
feature_names.dtypes
```

```
alcohol              float64
malic_acid           float64
ash                  float64
alcalinity_of_ash    float64
magnesium            float64
total_phenols        float64
dtype: object
```

```
X = np.asarray(feature_names) #independet variable independent
variable array got created
X[0:5] #show me elements from zeroth row to 5th row
```

```
array([[ 14.23,   1.71,   2.43,  15.6 , 127.  ,   2.8 ],
       [ 13.2 ,   1.78,   2.14,  11.2 , 100.  ,   2.65],
       [ 13.16,   2.36,   2.67,  18.6 , 101.  ,   2.8 ],
       [ 14.37,   1.95,   2.5 ,  16.8 , 113.  ,   3.85],
       [ 13.24,   2.59,   2.87,  21.  , 118.  ,   2.8 ]])
```

```
df['target_values'] = df['target_values'].astype('int')
Y = np.asarray(df['target_values']) #dependent variable
Y [0:5]
```

```
array([0, 0, 0, 0, 0])
```

Train/Test dataset

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=50)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (142, 6) (142,)
Test set: (36, 6) (36,)
```

Modeling (SVM with Scikit-learn)

```
from sklearn import svm
clf = svm.SVC(kernel='poly')
clf.fit(X_train, Y_train) # question and answers
```

```
SVC(kernel='poly')
```

```
yhat = clf.predict(X_test) #question
yhat [0:5]
```

```
array([1, 1, 1, 1, 1])
```

Evaluation (Using the different hyperparameters to find out the best accuracy)

```
from sklearn.metrics import f1_score
f1_score(Y_test, yhat, average='weighted')
```

```
0.48148148148148157
```

```
clf2 = svm.SVC(kernel='rbf')
clf2.fit(X_train, Y_train)
yhat2 = clf2.predict(X_test)
```

```
print("Avg F1-score: %.4f" % f1_score(Y_test, yhat2,
average="weighted"))
```

```
Avg F1-score: 0.4531
```

```
clf2 = svm.SVC(kernel='linear')
clf2.fit(X_train, Y_train)
yhat2 = clf2.predict(X_test)
print("Avg F1-score: %.4f" % f1_score(Y_test, yhat2,
average="weighted"))
```

```
Avg F1-score: 0.8889
```

```
df
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     target_values
0                0
1                0
2                0
3                0
4                0
..             ...
173              2
174              2
175              2
176              2
```

```
177              2

[178 rows x 7 columns]
```

```python
X_test
```

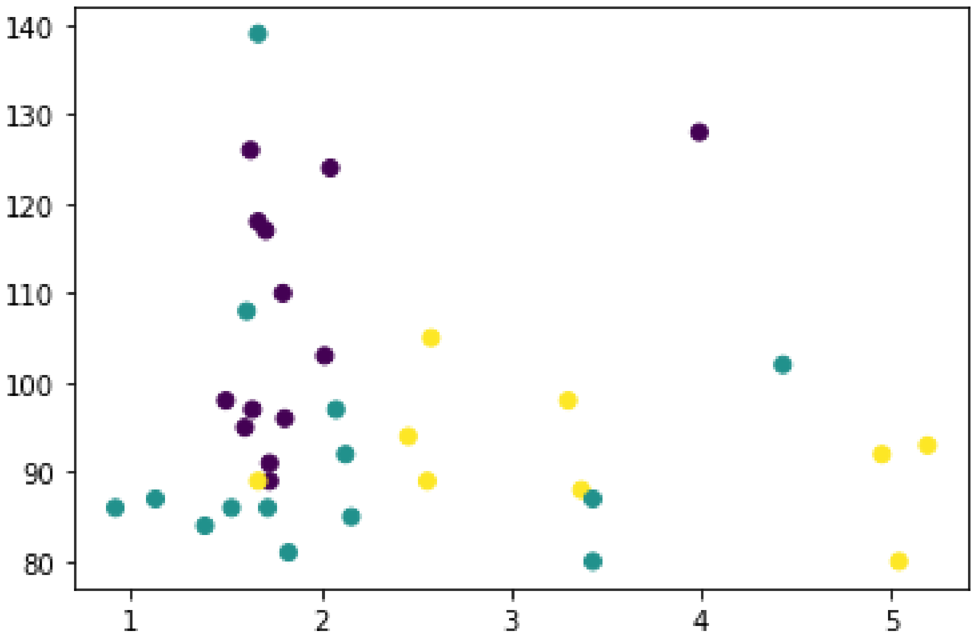
```
array([[ 12.42,   1.61,   2.19,  22.5 , 108.  ,   2.  ],
       [ 12.08,   1.83,   2.32,  18.5 ,  81.  ,   1.6 ],
       [ 12.  ,   0.92,   2.  ,  19.  ,  86.  ,   2.42],
       [ 13.36,   2.56,   2.35,  20.  ,  89.  ,   1.4 ],
       [ 13.62,   4.95,   2.35,  20.  ,  92.  ,   2.  ],
       [ 12.6 ,   2.46,   2.2 ,  18.5 ,  94.  ,   1.62],
       [ 12.42,   4.43,   2.73,  26.5 , 102.  ,   2.2 ],
       [ 12.77,   3.43,   1.98,  16.  ,  80.  ,   1.63],
       [ 13.5 ,   1.81,   2.61,  20.  ,  96.  ,   2.53],
       [ 12.82,   3.37,   2.3 ,  19.5 ,  88.  ,   1.48],
       [ 13.56,   1.71,   2.31,  16.2 , 117.  ,   3.15],
       [ 13.05,   2.05,   3.22,  25.  , 124.  ,   2.63],
       [ 12.  ,   3.43,   2.  ,  19.  ,  87.  ,   2.  ],
       [ 13.75,   1.73,   2.41,  16.  ,  89.  ,   2.6 ],
       [ 12.37,   1.13,   2.16,  19.  ,  87.  ,   3.5 ],
       [ 13.88,   5.04,   2.23,  20.  ,  80.  ,   0.98],
       [ 12.08,   1.39,   2.5 ,  22.5 ,  84.  ,   2.56],
       [ 13.48,   1.67,   2.64,  22.5 ,  89.  ,   2.6 ],
       [ 12.08,   2.08,   1.7 ,  17.5 ,  97.  ,   2.23],
       [ 13.74,   1.67,   2.25,  16.4 , 118.  ,   2.6 ],
       [ 12.85,   1.6 ,   2.52,  17.8 ,  95.  ,   2.48],
       [ 12.43,   1.53,   2.29,  21.5 ,  86.  ,   2.74],
       [ 13.07,   1.5 ,   2.1 ,  15.5 ,  98.  ,   2.4 ],
       [ 13.23,   3.3 ,   2.28,  18.5 ,  98.  ,   1.8 ],
       [ 13.51,   1.8 ,   2.65,  19.  , 110.  ,   2.35],
       [ 14.06,   1.63,   2.28,  16.  , 126.  ,   3.  ],
       [ 11.82,   1.72,   1.88,  19.5 ,  86.  ,   2.5 ],
       [ 12.99,   1.67,   2.6 ,  30.  , 139.  ,   3.3 ],
       [ 14.22,   3.99,   2.51,  13.2 , 128.  ,   3.  ],
       [ 11.79,   2.13,   2.78,  28.5 ,  92.  ,   2.13],
       [ 14.1 ,   2.02,   2.4 ,  18.8 , 103.  ,   2.75],
       [ 14.83,   1.64,   2.17,  14.  ,  97.  ,   2.8 ],
       [ 13.58,   2.58,   2.69,  24.5 , 105.  ,   1.55],
       [ 13.17,   5.19,   2.32,  22.  ,  93.  ,   1.74],
       [ 14.75,   1.73,   2.39,  11.4 ,  91.  ,   3.1 ],
       [ 12.07,   2.16,   2.17,  21.  ,  85.  ,   2.6 ]])
```

```python
X_test.shape
```

```
(36, 6)
```

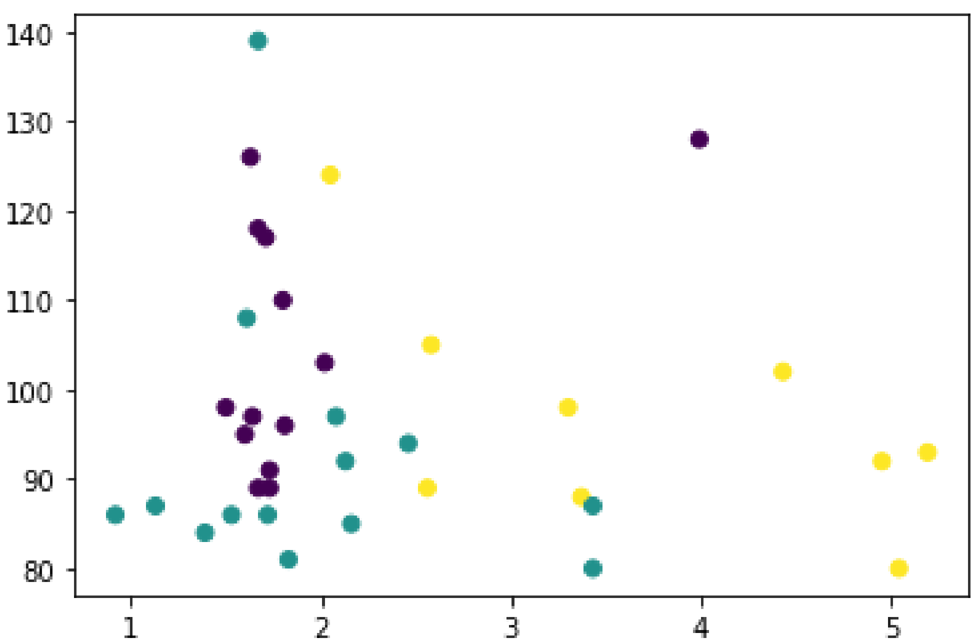
```python
import matplotlib.pyplot as plt
plt.scatter(X_test[:,1],X_test[:,-2],c=Y_test)
```

```
<matplotlib.collections.PathCollection at 0x155b48c2880>
```




```
import matplotlib.pyplot as plt
plt.scatter(X_test[:,1],X_test[:,-2],c=yhat2)
```

```
<matplotlib.collections.PathCollection at 0x155b492a9a0>
```




```
clf2.predict([[12.20,1.50,2.90,13.6,105,2.3]])
```

```
array([0])
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=50)
```

```
from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(X_train, Y_train) #Training Model
training_pred = clf.predict(X_train)
print(f"training accuracy is {f1_score(Y_train, training_pred, average='weighted')*100}")
```

```
training accuracy is 92.17933856828284
```

```
yhat = clf.predict(X_test) #final
from sklearn.metrics import f1_score

print(f"testing accuracy is {f1_score(Y_test, yhat, average='weighted') *100}")
```

```
testing accuracy is 88.88888888888889
```

Therefore the best training accuracy is 92.17 % and testing accuracy is 88.88% by using the hyperparameter as "linear"

The overall accuracy is 88.89%

Que.3 Using sklearn.datasets.load_diabetes apply Mutual info Regression and check which are the best columns according to the target column.

Then Apply decision tree on that data and try to get best accuracy by changing the hyperparameters

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.datasets import load_diabetes
```

```
diabetes = load_diabetes()
```

```
diabetes
```

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -
0.00259226,
          0.01990842, -0.01764613],
```

```
        [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
         -0.06832974, -0.09220405],
        [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
          0.00286377, -0.02593034],
        ...,
        [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
         -0.04687948,  0.01549073],
        [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
          0.04452837, -0.02593034],
        [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
         -0.00421986,  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 `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)',
 'feature_names': ['age',
  'sex',
  'bmi',
  'bp',
  's1',
  's2',
  's3',
  's4',
  's5',
  's6'],
 'data_filename': 'diabetes_data.csv.gz',
 'target_filename': 'diabetes_target.csv.gz',
 'data_module': 'sklearn.datasets.data'}
```

```
diabetes.keys() # Keys which we can use
```

```
dict_keys(['data', 'target', 'frame', 'DESCR', 'feature_names',
'data_filename', 'target_filename', 'data_module'])
```

```
diabetes.data # Keys which we can use
```

```
array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990842, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06832974, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286377, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04687948,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452837, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00421986,  0.00306441]])
```

```python
diabetes.target # Dependent Data
```

```
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.])
```

```
diabetes.feature_names # independent column names
```

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

```
diabetes.DESCR # Independent Column
```

```
'.. _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 `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)'
```

```python
print(diabetes["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 `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)
```

```python
#pd.DataFrame(Data,ColumnName)
df = pd.DataFrame(diabetes['data'],columns=diabetes['feature_names'])

df
```

```
          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6
0   -0.002592  0.019908 -0.017646
1   -0.039493 -0.068330 -0.092204
2   -0.002592  0.002864 -0.025930
3    0.034309  0.022692 -0.009362
4   -0.002592 -0.031991 -0.046641
..        ...       ...       ...
437 -0.002592  0.031193  0.007207
```

```
438  0.034309 -0.018118  0.044485
439 -0.011080 -0.046879  0.015491
440  0.026560  0.044528 -0.025930
441 -0.039493 -0.004220  0.003064

[442 rows x 10 columns]
```

```python
diabetes["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.])
```

```
df["target_values"] = diabetes.target # Dependent colummn CONTENT
```

```
df
```

```
          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 -0.005671 -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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6  target_values
0   -0.002592  0.019908 -0.017646          151.0
1   -0.039493 -0.068330 -0.092204           75.0
2   -0.002592  0.002864 -0.025930          141.0
3    0.034309  0.022692 -0.009362          206.0
4   -0.002592 -0.031991 -0.046641          135.0
..        ...       ...       ...            ...
437 -0.002592  0.031193  0.007207          178.0
438  0.034309 -0.018118  0.044485          104.0
439 -0.011080 -0.046879  0.015491          132.0
440  0.026560  0.044528 -0.025930          220.0
441 -0.039493 -0.004220  0.003064           57.0

[442 rows x 11 columns]
```

```
df.head()
```

```
        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 -0.005671 -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

         s4        s5        s6  target_values
```

```
0 -0.002592  0.019908 -0.017646          151.0
1 -0.039493 -0.068330 -0.092204           75.0
2 -0.002592  0.002864 -0.025930          141.0
3  0.034309  0.022692 -0.009362          206.0
4 -0.002592 -0.031991 -0.046641          135.0
```

```
df.tail()
```

```
          age       sex       bmi        bp        s1        s2
s3  \
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  0.017282 -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 -0.081414  0.083740  0.027809
0.173816

           s4        s5        s6  target_values
437 -0.002592  0.031193  0.007207          178.0
438  0.034309 -0.018118  0.044485          104.0
439 -0.011080 -0.046879  0.015491          132.0
440  0.026560  0.044528 -0.025930          220.0
441 -0.039493 -0.004220  0.003064           57.0
```

```
df.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_values  442 non-null    float64
dtypes: float64(11)
memory usage: 38.1 KB
```

```
df.describe()
```

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

                s2            s3            s4            s5
s6  \
count  4.420000e+02  4.420000e+02  4.420000e+02  4.420000e+02
4.420000e+02
mean   1.327024e-16 -4.574646e-16  3.777301e-16 -3.830854e-16 -
3.412882e-16
std    4.761905e-02  4.761905e-02  4.761905e-02  4.761905e-02
4.761905e-02
min   -1.156131e-01 -1.023071e-01 -7.639450e-02 -1.260974e-01 -
1.377672e-01
25%   -3.035840e-02 -3.511716e-02 -3.949338e-02 -3.324879e-02 -
3.317903e-02
50%   -3.819065e-03 -6.584468e-03 -2.592262e-03 -1.947634e-03 -
1.077698e-03
75%    2.984439e-02  2.931150e-02  3.430886e-02  3.243323e-02
2.791705e-02
max    1.987880e-01  1.811791e-01  1.852344e-01  1.335990e-01
1.356118e-01

       target_values
count     442.000000
mean      152.133484
std        77.093005
min        25.000000
25%        87.000000
50%       140.500000
75%       211.500000
max       346.000000
```

```
df.isnull().sum()
```

```
age              0
sex              0
bmi              0
bp               0
s1               0
s2               0
s3               0
s4               0
s5               0
s6               0
target_values    0
dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(df)
```

```
StandardScaler()
```

```
scaled_data = scaler.transform(df)
```

```
scaled_data
```

```
array([[ 0.80050009,  1.06548848,  1.29708846, ...,  0.41855058,
        -0.37098854, -0.01471948],
       [-0.03956713, -0.93853666, -1.08218016, ..., -1.43655059,
        -1.93847913, -1.00165882],
       [ 1.79330681,  1.06548848,  0.93453324, ...,  0.06020733,
        -0.54515416, -0.14457991],
       ...,
       [ 0.87686984,  1.06548848, -0.33441002, ..., -0.98558469,
         0.32567395, -0.26145431],
       [-0.9560041 , -0.93853666,  0.82123474, ...,  0.93615545,
        -0.54515416,  0.88131756],
       [-0.9560041 , -0.93853666, -1.53537419, ..., -0.08871747,
         0.06442552, -1.23540761]])
```

```
scaled_data.shape
```

```
(442, 11)
```

```
### Train test split to avoid overfitting
from sklearn.model_selection import train_test_split
X = df.drop(['target_values'],axis = 1) #independent var
Y = df['target_values'] #dependent var

X_train,X_test,Y_train,Y_test=train_test_split(X, #INDEPENDENDENT
VARIABLE
    Y, #wine as DEPENDENT VARIABLE
    test_size=0.3, #70% TRAINING DS AND 30% TEST DATA
    random_state=0)
```

```
X_train.shape
```

```
(309, 10)
```

```
from sklearn.feature_selection import mutual_info_regression
# determine the mutual information
mutual_info = mutual_info_regression(X_train, Y_train)
mutual_info #impactful variable will get high value and less
impactfull will get low values
```

```
array([0.00980229, 0.02745252, 0.18930309, 0.10498738, 0.08096084,
       0.00332908, 0.06218525, 0.11967625, 0.15140476, 0.14339862])
```

```
len(mutual_info)
```

```
10
```

```
mutual_info
```

```
array([0.00980229, 0.02745252, 0.18930309, 0.10498738, 0.08096084,
       0.00332908, 0.06218525, 0.11967625, 0.15140476, 0.14339862])
```

```
X_train.columns
```

```
Index(['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'],
dtype='object')
```

```
mutual_info
```

```
array([0.00980229, 0.02745252, 0.18930309, 0.10498738, 0.08096084,
       0.00332908, 0.06218525, 0.11967625, 0.15140476, 0.14339862])
```

```
mutual_info = pd.Series(mutual_info)
```

```
mutual_info
```

```
0    0.009802
1    0.027453
2    0.189303
3    0.104987
4    0.080961
5    0.003329
6    0.062185
7    0.119676
8    0.151405
9    0.143399
dtype: float64
```

```
type(mutual_info)
```

```
pandas.core.series.Series
```

```
mutual_info.index
```

```
RangeIndex(start=0, stop=10, step=1)
```

```
mutual_info.index = X_train.columns
```

```
mutual_info
```

```
age    0.009802
sex    0.027453
bmi    0.189303
bp     0.104987
s1     0.080961
s2     0.003329
s3     0.062185
s4     0.119676
s5     0.151405
s6     0.143399
dtype: float64
```

```
mutual_info.sort_values(ascending=False)
```

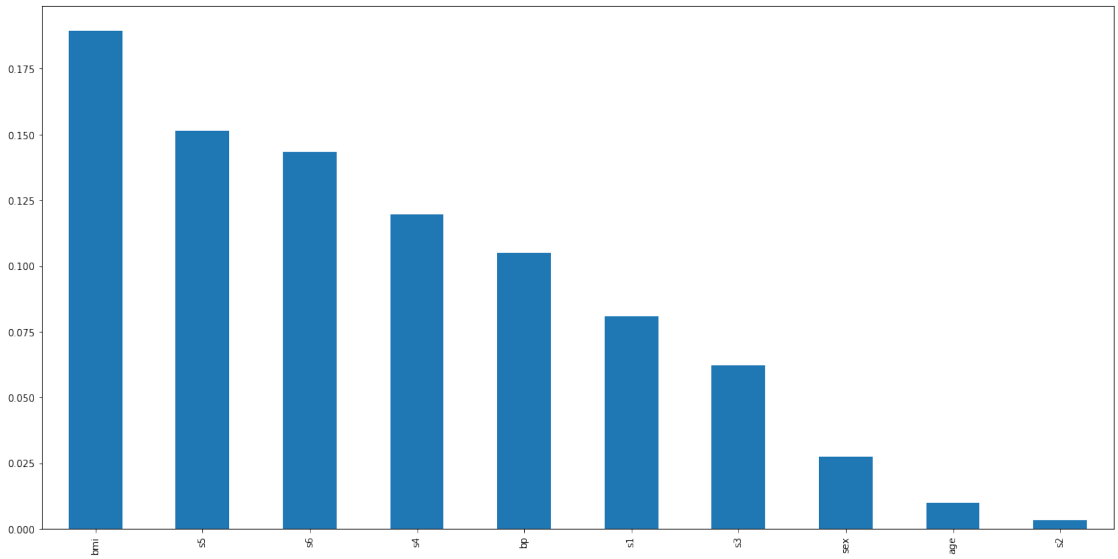
```
bmi    0.189303
s5     0.151405
s6     0.143399
s4     0.119676
bp     0.104987
s1     0.080961
s3     0.062185
sex    0.027453
age    0.009802
s2     0.003329
dtype: float64
```

```
mutual_info.index[0]
```

```
'age'
```

```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(20,10))
```

```
<AxesSubplot:>
```




```
mutual_info[0]
```

```
0.009802289238992401
```

```
mutual_info
```

```
age    0.009802
sex    0.027453
bmi    0.189303
bp     0.104987
s1     0.080961
s2     0.003329
s3     0.062185
s4     0.119676
s5     0.151405
s6     0.143399
dtype: float64
```

```
Y_train.shape
```

```
(309,)
```

```
def select_columns(MI,threshold):
    columns = []
    for i in range(len(MI)):#0-36
        if MI[i] > threshold:
            columns.append(MI.index[i])
    print(columns)
    return columns
```

```
good_columns = select_columns(mutual_info,0.1)
```

```
['bmi', 'bp', 's4', 's5', 's6']
```

```
X_train
```

```
          age       sex       bmi        bp        s1        s2
s3  \
232  0.012648  0.050680  0.000261 -0.011409  0.039710  0.057245 -
0.039719
224 -0.027310 -0.044642 -0.066563 -0.112400 -0.049727 -0.041397
0.000779
252  0.005383 -0.044642  0.059541 -0.056166  0.024574  0.052861 -
0.043401
254  0.030811  0.050680  0.056307  0.076958  0.049341 -0.012274 -
0.036038
418  0.009016 -0.044642 -0.024529 -0.026328  0.098876  0.094196
0.070730
..        ...       ...       ...       ...       ...       ...
...
323  0.070769  0.050680 -0.007284  0.049415  0.060349 -0.004445 -
0.054446
192  0.056239  0.050680 -0.030996  0.008101  0.019070  0.021233
0.033914
117  0.059871 -0.044642 -0.021295  0.087287  0.045213  0.031567 -
0.047082
47  -0.078165 -0.044642 -0.073030 -0.057314 -0.084126 -0.074277 -
0.024993
172  0.041708  0.050680  0.071397  0.008101  0.038334  0.015909 -
0.017629

           s4        s5        s6
232  0.056081  0.024053  0.032059
224 -0.039493 -0.035817 -0.009362
252  0.050914 -0.004220 -0.030072
254  0.071210  0.120053  0.090049
418 -0.002592 -0.021394  0.007207
..        ...       ...       ...
323  0.108111  0.129019  0.056912
192 -0.039493 -0.029528 -0.059067
117  0.071210  0.079121  0.135612
47  -0.039493 -0.018118 -0.083920
172  0.034309  0.073410  0.085907

[309 rows x 10 columns]
```

```
X_train_columns = X_train[good_columns]
```

```
X_train_columns
```

```
          bmi        bp        s4        s5        s6
232  0.000261 -0.011409  0.056081  0.024053  0.032059
224 -0.066563 -0.112400 -0.039493 -0.035817 -0.009362
252  0.059541 -0.056166  0.050914 -0.004220 -0.030072
254  0.056307  0.076958  0.071210  0.120053  0.090049
418 -0.024529 -0.026328 -0.002592 -0.021394  0.007207
..        ...       ...       ...       ...       ...
```

```
323 -0.007284  0.049415  0.108111  0.129019  0.056912
192 -0.030996  0.008101 -0.039493 -0.029528 -0.059067
117 -0.021295  0.087287  0.071210  0.079121  0.135612
47  -0.073030 -0.057314 -0.039493 -0.018118 -0.083920
172  0.071397  0.008101  0.034309  0.073410  0.085907

[309 rows x 5 columns]
```

Using the Decision Tree Model

```
X = X_train[good_columns]   # Independent Variable
```

```
X
```

```
          bmi        bp        s4        s5        s6
232  0.000261 -0.011409  0.056081  0.024053  0.032059
224 -0.066563 -0.112400 -0.039493 -0.035817 -0.009362
252  0.059541 -0.056166  0.050914 -0.004220 -0.030072
254  0.056307  0.076958  0.071210  0.120053  0.090049
418 -0.024529 -0.026328 -0.002592 -0.021394  0.007207
..        ...       ...       ...       ...       ...
323 -0.007284  0.049415  0.108111  0.129019  0.056912
192 -0.030996  0.008101 -0.039493 -0.029528 -0.059067
117 -0.021295  0.087287  0.071210  0.079121  0.135612
47  -0.073030 -0.057314 -0.039493 -0.018118 -0.083920
172  0.071397  0.008101  0.034309  0.073410  0.085907

[309 rows x 5 columns]
```

```
X.shape
```

```
(309, 5)
```

```
Y = df['target_values']
```

```
Y
```

```
0      0.0
1      NaN
2      0.0
3      0.0
4      NaN
      ...
437    NaN
438    NaN
439    0.0
440    0.0
441    NaN
Name: target_values, Length: 442, dtype: float64
```

```
Y.isnull().sum()
```

```
133
```

```python
Y1 = Y.dropna()
```

```python
Y1
```

```
0      0.0
2      0.0
3      0.0
9      0.0
11     0.0
      ...
432    0.0
433    0.0
436    0.0
439    0.0
440    0.0
Name: target_values, Length: 309, dtype: float64
```

```python
Y1.shape
```

```
(309,)
```

Setting up Decision Tree

```python
from sklearn.model_selection import train_test_split
```

```python
X_train, X_test, Y1_train, Y1_test = train_test_split(X, Y1,
test_size=0.3, random_state=0)
```

```python
X_train
```

```
          bmi        bp        s4        s5        s6
32   0.125287  0.028758  0.108111  0.000271  0.027917
279 -0.024529  0.004658 -0.039493 -0.015998 -0.025930
420 -0.036385  0.000068  0.034309 -0.033249  0.061054
41  -0.067641 -0.108957 -0.039493 -0.049868 -0.009362
277 -0.059019  0.001215 -0.076395 -0.021394  0.015491
..        ...       ...       ...       ...       ...
385 -0.019140  0.049415 -0.039493 -0.025952 -0.013504
222 -0.025607  0.042530 -0.039493  0.001144  0.019633
293  0.092953  0.012691  0.000360 -0.054544 -0.001078
367  0.170555  0.014987  0.034309  0.033657  0.032059
182  0.005650  0.056301  0.071210  0.015567 -0.009362

[216 rows x 5 columns]
```

```python
X_test
```

```
          bmi        bp        s4        s5        s6
34  -0.063330 -0.057314 -0.039493 -0.059473 -0.067351
121  0.017506  0.021872  0.034309  0.019908  0.011349
```

```
354  0.045529  0.021872  0.034309  0.074193  0.061054
340 -0.013751  0.132044 -0.039493 -0.035817 -0.030072
351 -0.040696 -0.033214 -0.039493 -0.057800 -0.042499
..        ...       ...       ...       ...       ...
226 -0.046085 -0.026328 -0.039493 -0.039810 -0.054925
398  0.015350 -0.033214 -0.002592  0.045066 -0.067351
30   0.044451 -0.019442 -0.039493 -0.027129 -0.009362
431 -0.030996  0.021872 -0.039493 -0.014956 -0.001078
258 -0.024529 -0.042395  0.080804 -0.037128  0.056912

[93 rows x 5 columns]
```

```
Y1_train
```

```
372    0.0
306    0.0
241    0.0
258    0.0
425    0.0
      ...
355    0.0
269    0.0
174    0.0
77     0.0
245    0.0
Name: target_values, Length: 216, dtype: float64
```

```
Y1_test
```

```
97     0.0
324    0.0
239    0.0
229    0.0
265    0.0
      ...
228    0.0
379    0.0
58     0.0
195    0.0
250    0.0
Name: target_values, Length: 93, dtype: float64
```

Modelling

```
from sklearn.tree import DecisionTreeRegressor
target_values = DecisionTreeRegressor(criterion = "squared_error",
max_depth = 4)
target_values # it shows the default parameters
```

```
DecisionTreeRegressor(max_depth=4)
```

```
target_values.fit(X_train,Y1_train)
```

```
DecisionTreeRegressor(max_depth=4)
```

```
y_pred = target_values.predict(X_test)
```

```
y_pred
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.,
       0., 0., 0., 0., 0., 0., 0., 0.])
```

```
X_test.shape
```

```
(93, 5)
```

```
Y1_test.shape
```

```
(93,)
```

```
y_pred.shape
```

```
(93,)
```

```
print (y_pred [0:5])#predicted by the ml model
print (Y1_test [0:5])#actual values we have
```

```
[0. 0. 0. 0. 0.]
97     0.0
324    0.0
239    0.0
229    0.0
265    0.0
Name: target_values, dtype: float64
```

```
Y1_test
```

```
97     0.0
324    0.0
239    0.0
229    0.0
265    0.0
      ...
228    0.0
379    0.0
```

```
58     0.0
195    0.0
250    0.0
Name: target_values, Length: 93, dtype: float64
```

```
from sklearn.metrics import r2_score
```

```
Y1_test.shape
```

```
(93,)
```

```
y_pred.shape
```

```
(93,)
```

```
print(f"R2 Score : {r2_score(Y1_test,y_pred)*100} % ")
```

```
R2 Score : 100.0 %
```

```
accuracy = (f"accuracy : {r2_score(Y1_test,y_pred)*100} % ")
```

```
accuracy
```

```
'accuracy : 100.0 % '
```

Que 4 Using sklearn.datasets.load_boston apply Mutual info Regression and check which are the best columns according to the target column.

Then Apply MultiLinear Regression on that data and try to get best accuracy by changing the hyperparameters

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.datasets import load_boston
```

```
boston_df = load_boston()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\utils\
deprecation.py:87: FutureWarning: Function load_boston is deprecated;
`load_boston` is deprecated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can
refer to
```

```
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use
of this
    dataset unless the purpose of the code is to study and educate
about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22,
header=None)
        data = np.hstack([raw_df.values[::2, :],
raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames
housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

```
boston_df
```

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01,
3.9690e+02,
         4.9800e+00],
        [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01,
3.9690e+02,
         9.1400e+00],
        [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01,
3.9283e+02,
         4.0300e+00],
```

```
        ...,
        [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9690e+02,
         5.6400e+00],
        [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9345e+02,
         6.4800e+00],
        [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9690e+02,
         7.8800e+00]]),
 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1,
16.5, 18.9, 15. ,
        18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6,
19.6,
        15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5,
13.2,
        13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3,
24.7,
        21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4,
18.9,
        35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. ,
23.5,
        19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4,
20. ,
        20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5,
22.2,
        23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7,
43.8,
        33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8,
19.4,
        21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3,
22. ,
        20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2,
19.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4,
13.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3,
19.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. ,
22.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6,
29.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6,
50. ,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3,
30.3,
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5,
24.4,
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5,
23. ,
```

```
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5,
24.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. ,
20.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8,
29.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8,
31. ,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2,
32.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2,
22. ,
        20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6,
27.1,
        20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4,
28.2,
        22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8,
23.1,
        21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3,
22.6,
        19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. ,
18.7,
        32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9,
24.1,
        18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9,
20.8,
        16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. ,
13.8,
        13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,
8.8,
         7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7,
13.1,
        12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. ,
11.9,
        27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5,
10.4,
         8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9,
11. ,
         9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4,
12.8,
        10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. ,
13.4,
        15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4,
17.7,
        19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6,
23.2,
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. ,
21.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8,
24.5,
```

```
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. ,
11.9]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM',
'AGE', 'DIS', 'RAD',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\
n---------------------------\n\n**Data Set Characteristics:**  \n\n
:Number of Instances: 506 \n\n    :Number of Attributes: 13
numeric/categorical predictive. Median Value (attribute 14) is usually
the target.\n\n    :Attribute Information (in order):\n        - CRIM
per capita crime rate by town\n        - ZN       proportion of
residential land zoned for lots over 25,000 sq.ft.\n        - INDUS
proportion of non-retail business acres per town\n        - CHAS
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\
n        - NOX      nitric oxides concentration (parts per 10
million)\n        - RM       average number of rooms per dwelling\n
- AGE      proportion of owner-occupied units built prior to 1940\n
- DIS      weighted distances to five Boston employment centres\n
- RAD      index of accessibility to radial highways\n        - TAX
full-value property-tax rate per $10,000\n        - PTRATIO  pupil-
teacher ratio by town\n        - B        1000(Bk - 0.63)^2 where Bk
is the proportion of black people by town\n        - LSTAT    % lower
status of the population\n        - MEDV     Median value of owner-
occupied homes in $1000's\n\n    :Missing Attribute Values: None\n\n
:Creator: Harrison, D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML
housing dataset.\nhttps://archive.ics.uci.edu/ml/machine-learning-
databases/housing/\n\n\nThis dataset was taken from the StatLib
library which is maintained at Carnegie Mellon University.\n\nThe
Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\
nprices and the demand for clean air', J. Environ. Economics &
Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch,
'Regression diagnostics\n...', Wiley, 1980.   N.B. Various
transformations are used in the table on\npages 244-261 of the
latter.\n\nThe Boston house-price data has been used in many machine
learning papers that address regression\nproblems.   \n     \n..
topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression
diagnostics: Identifying Influential Data and Sources of
Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993).
Combining Instance-Based and Model-Based Learning. In Proceedings on
the Tenth International Conference of Machine Learning, 236-243,
University of Massachusetts, Amherst. Morgan Kaufmann.\n",
 'filename': 'boston_house_prices.csv',
 'data_module': 'sklearn.datasets.data'}
```

```
boston_df.data
```

```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01,
3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01,
3.9690e+02,
```

```
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01,
3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01,
3.9690e+02,
        7.8800e+00]])
```

```
boston_df.keys()
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename',
'data_module'])
```

```
boston_df.feature_names
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

```
boston_df.target # Dependent Data
```

```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15.
,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6,
19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5,
13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3,
24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4,
18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. ,
23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20.
,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5,
22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7,
43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8,
19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22.
,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2,
```

```
19.6,
        23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4,
13.4,
        15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3,
19.4,
        17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. ,
22.7,
        25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6,
29.4,
        23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50.
,
        32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3,
30.3,
        34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5,
24.4,
        20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23.
,
        26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5,
24.3,
        31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. ,
20.1,
        22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8,
29.6,
        42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31.
,
        36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2,
32.4,
        32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22.
,
        20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6,
27.1,
        20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4,
28.2,
        22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8,
23.1,
        21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3,
22.6,
        19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. ,
18.7,
        32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9,
24.1,
        18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9,
20.8,
        16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. ,
13.8,
        13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,
8.8,
         7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7,
13.1,
        12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. ,
```

```
11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5,
10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11.
,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4,
12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. ,
13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4,
17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6,
23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. ,
21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8,
24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. ,
11.9])
```

```
#pd.DataFrame(Data,ColumnName)
df =
pd.DataFrame(boston_df['data'],columns=boston_df['feature_names'])
```

```
df
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD
TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0
296.0
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0
242.0
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0
242.0
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0
222.0
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0
222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ..
.
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0
273.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0
273.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0
273.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0
273.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0
273.0
```

```
     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
```

```
df["target_values"] = boston_df.target # Dependent colummn CONTENT
```

```
df
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD
TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0
296.0
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0
242.0
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0
242.0
3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0
222.0
4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0
222.0
..       ...   ...    ...   ...    ...    ...   ...     ...  ...    ..
.
501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0
273.0
502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0
273.0
503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0
273.0
504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0
273.0
505  0.04741   0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0
273.0

     PTRATIO       B  LSTAT  target_values
0       15.3  396.90   4.98           24.0
1       17.8  396.90   9.14           21.6
2       17.8  392.83   4.03           34.7
3       18.7  394.63   2.94           33.4
4       18.7  396.90   5.33           36.2
..       ...     ...    ...            ...
```

```
501     21.0  391.99   9.67           22.4
502     21.0  396.90   9.08           20.6
503     21.0  396.90   5.64           23.9
504     21.0  393.45   6.48           22.0
505     21.0  396.90   7.88           11.9

[506 rows x 14 columns]
```

df.head()

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX
\
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0

1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0

2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0

3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0

4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0


   PTRATIO       B  LSTAT  target_values
0     15.3  396.90   4.98           24.0
1     17.8  396.90   9.14           21.6
2     17.8  392.83   4.03           34.7
3     18.7  394.63   2.94           33.4
4     18.7  396.90   5.33           36.2
```

df.tail()

```
        CRIM   ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX
\
501  0.06263  0.0  11.93   0.0  0.573  6.593  69.1  2.4786  1.0  273.0

502  0.04527  0.0  11.93   0.0  0.573  6.120  76.7  2.2875  1.0  273.0

503  0.06076  0.0  11.93   0.0  0.573  6.976  91.0  2.1675  1.0  273.0

504  0.10959  0.0  11.93   0.0  0.573  6.794  89.3  2.3889  1.0  273.0

505  0.04741  0.0  11.93   0.0  0.573  6.030  80.8  2.5050  1.0  273.0


     PTRATIO       B  LSTAT  target_values
501     21.0  391.99   9.67           22.4
502     21.0  396.90   9.08           20.6
503     21.0  396.90   5.64           23.9
```

```
504     21.0  393.45   6.48           22.0
505     21.0  396.90   7.88           11.9
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   CRIM           506 non-null    float64
 1   ZN             506 non-null    float64
 2   INDUS          506 non-null    float64
 3   CHAS           506 non-null    float64
 4   NOX            506 non-null    float64
 5   RM             506 non-null    float64
 6   AGE            506 non-null    float64
 7   DIS            506 non-null    float64
 8   RAD            506 non-null    float64
 9   TAX            506 non-null    float64
 10  PTRATIO        506 non-null    float64
 11  B              506 non-null    float64
 12  LSTAT          506 non-null    float64
 13  target_values  506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

```
df.isnull().sum()
```

```
CRIM             0
ZN               0
INDUS            0
CHAS             0
NOX              0
RM               0
AGE              0
DIS              0
RAD              0
TAX              0
PTRATIO          0
B                0
LSTAT            0
target_values    0
dtype: int64
```

```
df.describe()
```

```
             CRIM          ZN       INDUS        CHAS         NOX
RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
```

```
mean      3.613524   11.363636   11.136779    0.069170    0.554695
6.284634
std       8.601545   23.322453    6.860353    0.253994    0.115878
0.702617
min       0.006320    0.000000    0.460000    0.000000    0.385000
3.561000
25%       0.082045    0.000000    5.190000    0.000000    0.449000
5.885500
50%       0.256510    0.000000    9.690000    0.000000    0.538000
6.208500
75%       3.677083   12.500000   18.100000    0.000000    0.624000
6.623500
max      88.976200  100.000000   27.740000    1.000000    0.871000
8.780000

              AGE         DIS         RAD         TAX     PTRATIO
B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534
356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946
91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000
0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000
375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000
391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000
396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000
396.900000

            LSTAT  target_values
count  506.000000     506.000000
mean    12.653063      22.532806
std      7.141062       9.197104
min      1.730000       5.000000
25%      6.950000      17.025000
50%     11.360000      21.200000
75%     16.955000      25.000000
max     37.970000      50.000000
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df)
```

```
StandardScaler()
```

```
scaled_data = scaler.transform(df)

scaled_data

array([[-0.41978194,  0.28482986, -1.2879095 , ...,  0.44105193,
        -1.0755623 ,  0.15968566],
       [-0.41733926, -0.48772236, -0.59338101, ...,  0.44105193,
        -0.49243937, -0.10152429],
       [-0.41734159, -0.48772236, -0.59338101, ...,  0.39642699,
        -1.2087274 ,  1.32424667],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  0.44105193,
        -0.98304761,  0.14880191],
       [-0.40776407, -0.48772236,  0.11573841, ...,  0.4032249 ,
        -0.86530163, -0.0579893 ],
       [-0.41500016, -0.48772236,  0.11573841, ...,  0.44105193,
        -0.66905833, -1.15724782]])

scaled_data.shape

(506, 14)
```

Train and Test Split

```
### Train test split to avoid overfitting
from sklearn.model_selection import train_test_split
X = df.drop(['target_values'],axis = 1) #independent var
Y = df['target_values'] #dependent var

X_train,X_test,Y_train,Y_test=train_test_split(X, #INDEPENDENDENT
VARIABLE
    Y, # target_values as DEPENDENT VARIABLE
    test_size=0.3, #70% TRAINING DS AND 30% TEST DATA
    random_state=100)

from sklearn.feature_selection import mutual_info_regression
# determine the mutual information
mutual_info = mutual_info_regression(X_train, Y_train)
mutual_info #impactful variable will get high value and less
impactfull will get low values

array([0.35174779, 0.17509246, 0.43835523, 0.02526545, 0.37391798,
       0.52100482, 0.28793572, 0.29695079, 0.19063316, 0.29278084,
       0.33842663, 0.13964175, 0.7318488 ])

mutual_info

array([0.35174779, 0.17509246, 0.43835523, 0.02526545, 0.37391798,
       0.52100482, 0.28793572, 0.29695079, 0.19063316, 0.29278084,
       0.33842663, 0.13964175, 0.7318488 ])

X_train.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
```

```
mutual_info
```

```
array([0.35174779, 0.17509246, 0.43835523, 0.02526545, 0.37391798,
       0.52100482, 0.28793572, 0.29695079, 0.19063316, 0.29278084,
       0.33842663, 0.13964175, 0.7318488 ])
```

```
mutual_info = pd.Series(mutual_info)
```

```
mutual_info
```

```
0     0.351748
1     0.175092
2     0.438355
3     0.025265
4     0.373918
5     0.521005
6     0.287936
7     0.296951
8     0.190633
9     0.292781
10    0.338427
11    0.139642
12    0.731849
dtype: float64
```

```
X_train.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT'],
      dtype='object')
```

```
type(mutual_info)
```

```
pandas.core.series.Series
```

```
mutual_info.index
```

```
RangeIndex(start=0, stop=13, step=1)
```

```
mutual_info.index = X_train.columns
```

```
mutual_info
```

```
CRIM       0.351748
ZN         0.175092
INDUS      0.438355
CHAS       0.025265
```

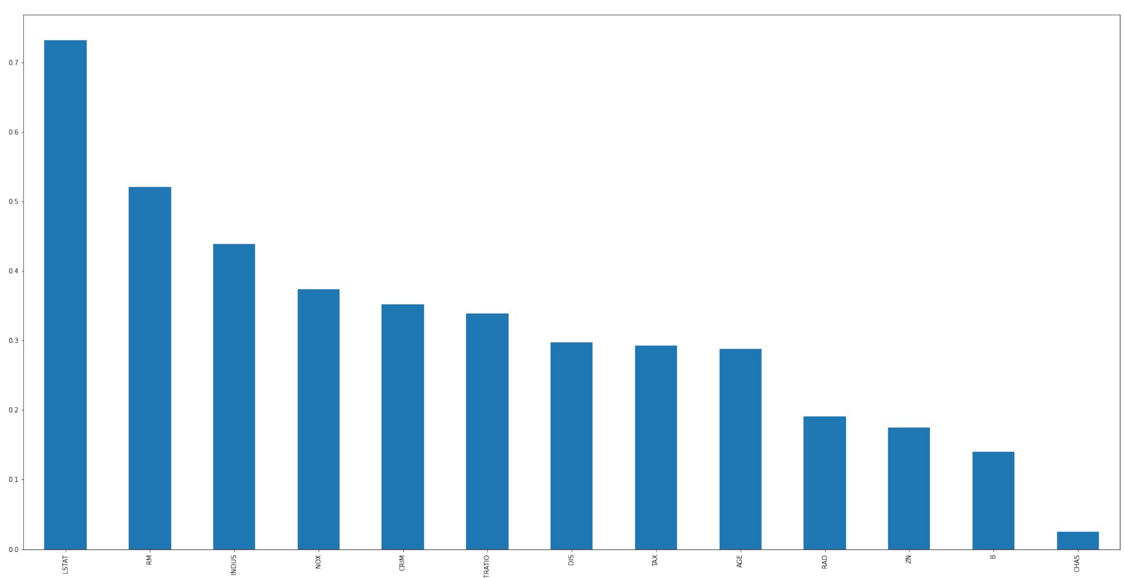
```
NOX        0.373918
RM         0.521005
AGE        0.287936
DIS        0.296951
RAD        0.190633
TAX        0.292781
PTRATIO    0.338427
B          0.139642
LSTAT      0.731849
dtype: float64
```

```
mutual_info.sort_values(ascending=False)
```

```
LSTAT      0.731849
RM         0.521005
INDUS      0.438355
NOX        0.373918
CRIM       0.351748
PTRATIO    0.338427
DIS        0.296951
TAX        0.292781
AGE        0.287936
RAD        0.190633
ZN         0.175092
B          0.139642
CHAS       0.025265
dtype: float64
```

```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(30,15))
```

```
<AxesSubplot:>
```




```
mutual_info[0]
```

```
0.35174779022450053
```

```
mutual_info
```

```
CRIM       0.351748
ZN         0.175092
INDUS      0.438355
CHAS       0.025265
NOX        0.373918
RM         0.521005
AGE        0.287936
DIS        0.296951
RAD        0.190633
TAX        0.292781
PTRATIO    0.338427
B          0.139642
LSTAT      0.731849
dtype: float64
```

```
def select_columns(MI,threshold):
    columns = []
    for i in range(len(MI)):#0-36
        if MI[i] > threshold:
            columns.append(MI.index[i])
    print(columns)
    return columns
```

```
good_columns = select_columns(mutual_info,0.30)
```

```
['CRIM', 'INDUS', 'NOX', 'RM', 'PTRATIO', 'LSTAT']
```

```
X_train_columns = X_train[good_columns]
```

```
X_train_columns
```

```
         CRIM  INDUS     NOX     RM  PTRATIO  LSTAT
463   5.82115  18.10  0.7130  6.513     20.2  10.29
75    0.09512  12.83  0.4370  6.286     18.7   8.94
478  10.23300  18.10  0.6140  6.185     20.2  18.03
199   0.03150   1.47  0.4030  6.975     17.0   4.56
84    0.05059   4.49  0.4490  6.389     18.5   9.62
..        ...    ...     ...    ...      ...    ...
343   0.02543   3.78  0.4840  6.696     17.6   7.18
359   4.26131  18.10  0.7700  6.112     20.2  12.67
323   0.28392   7.38  0.4930  5.708     19.6  11.74
280   0.03578   3.33  0.4429  7.820     14.9   3.76
8     0.21124   7.87  0.5240  5.631     15.2  29.93

[354 rows x 6 columns]
```

```
X = df[["CRIM","INDUS","NOX","RM","PTRATIO","LSTAT"]].values  # 
Independent Variable
```

```
X
```

```
array([[6.3200e-03, 2.3100e+00, 5.3800e-01, 6.5750e+00, 1.5300e+01,
         4.9800e+00],
        [2.7310e-02, 7.0700e+00, 4.6900e-01, 6.4210e+00, 1.7800e+01,
         9.1400e+00],
        [2.7290e-02, 7.0700e+00, 4.6900e-01, 7.1850e+00, 1.7800e+01,
         4.0300e+00],
        ...,
        [6.0760e-02, 1.1930e+01, 5.7300e-01, 6.9760e+00, 2.1000e+01,
         5.6400e+00],
        [1.0959e-01, 1.1930e+01, 5.7300e-01, 6.7940e+00, 2.1000e+01,
         6.4800e+00],
        [4.7410e-02, 1.1930e+01, 5.7300e-01, 6.0300e+00, 2.1000e+01,
         7.8800e+00]])
```

```
X.shape
```

```
(506, 6)
```

```
Y = df[["target_values"]] # Dependent Variable
```

```
Y.shape
```

```
(506, 1)
```

```
Y.isnull().sum()
```

```
target_values    0
dtype: int64
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 
0.2, random_state = 100)
```

```
X_train.shape
```

```
(404, 6)
```

```
X_test.shape
```

```
(102, 6)
```

Feature Scaling

```
# Feature Scaling

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_Y = StandardScaler()
```

```
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
Y_train = sc_Y.fit_transform(Y_train)
Y_test = sc_Y.transform(Y_test)

X_train

array([[ 1.89243384,  1.03527975,  1.01954892, -0.07476467,
0.81397465,
         1.26267819],
       [-0.32767247, -0.16180187, -0.07738714, -0.21818518, -
0.03955436,
        -0.91004324],
       [-0.27135559,  1.25133839,  0.44948766,  0.94763894, -
1.79403066,
        -1.10118772],
       ...,
       [-0.39352177, -0.52968549, -0.51788902, -0.80606725,
0.52946498,
        -0.11796252],
       [-0.42578271, -1.12092703, -0.95061734,  2.19298334, -1.6991941
,
        -1.21532436],
       [-0.40297097, -0.45815257, -0.25013298, -0.91540764, -
1.55693926,
         2.38341741]])

Y_train

array([[-1.37750661e+00],
       [-5.77838509e-02],
       [ 2.07151674e+00],
       [-5.56838677e-01],
       [ 2.30558938e-01],
       [-1.56603844e+00],
       [-1.61039887e+00],
       [ 4.19090761e-01],
       [ 4.20271144e-02],
       [ 1.38393009e+00],
       [ 1.41720041e+00],
       [-1.08916383e+00],
       [ 1.52928187e-01],
       [-2.68495889e-01],
       [ 7.51793978e-01],
       [-7.34280393e-01],
       [ 3.03635607e+00],
       [-1.56603844e+00],
       [-8.56271573e-01],
       [-9.78262753e-01],
       [-9.10541726e-02],
       [ 5.31172216e-02],
```

```
[-2.79585996e-01],
[-1.02262318e+00],
[-8.11911144e-01],
[-2.90676103e-01],
[-2.90676103e-01],
[-3.23946425e-01],
[ 1.98468999e-02],
[-4.90298034e-01],
[ 7.52974361e-02],
[-4.68117819e-01],
[ 4.20271144e-02],
[ 3.03635607e+00],
[ 5.31172216e-02],
[ 7.52974361e-02],
[-1.35414602e-01],
[-1.35414602e-01],
[-1.05589350e+00],
[-1.73239005e+00],
[-9.22812216e-01],
[ 9.51415909e-01],
[ 8.73785158e-01],
[-3.35036532e-01],
[ 3.03635607e+00],
[ 1.97288616e-01],
[-4.23757390e-01],
[ 8.07244515e-01],
[-1.11134404e+00],
[ 2.59275178e+00],
[-2.68495889e-01],
[-1.35414602e-01],
[-8.89541895e-01],
[ 1.36174988e+00],
[ 1.41838080e-01],
[ 6.42073289e-02],
[-6.45559535e-01],
[-7.67550715e-01],
[-2.24135460e-01],
[ 9.95776338e-01],
[ 1.19657865e-01],
[-6.01199106e-01],
[ 1.15103784e+00],
[ 6.74163228e-01],
[-7.99640654e-02],
[ 2.86009474e-01],
[-2.24135460e-01],
[-7.12100179e-01],
[ 6.74163228e-01],
[-6.88739581e-02],
[-4.01577176e-01],
[ 1.36174988e+00],
```

```
[ 5.96532477e-01],
[-2.33331464e-03],
[-2.33331464e-03],
[-5.34658463e-01],
[ 7.07433549e-01],
[-3.68306854e-01],
[ 1.59464213e+00],
[-6.12289213e-01],
[-4.45937605e-01],
[ 1.98468999e-02],
[-3.12856318e-01],
[-8.67361680e-01],
[-1.02144280e-01],
[ 1.86198509e-01],
[ 2.31549910e+00],
[-2.57405781e-01],
[ 4.19090761e-01],
[-2.01955245e-01],
[-9.78262753e-01],
[-7.23190286e-01],
[ 1.00686645e+00],
[-7.78640822e-01],
[-9.11722109e-01],
[-1.95419219e+00],
[ 3.03635607e+00],
[-6.89919964e-01],
[-2.90676103e-01],
[-8.23001251e-01],
[-4.66937436e-02],
[ 2.74919367e-01],
[-3.79396961e-01],
[ 1.04013677e+00],
[-3.01766210e-01],
[-9.44992431e-01],
[-4.34847497e-01],
[-1.14461436e+00],
[-3.56036364e-02],
[ 9.29235694e-01],
[-2.35225567e-01],
[ 7.52974361e-02],
[-1.02262318e+00],
[-9.67172645e-01],
[ 1.08567758e-01],
[-6.67739750e-01],
[-4.57027712e-01],
[-3.57216747e-01],
[-1.18897479e+00],
[-3.35036532e-01],
[-3.35036532e-01],
[ 1.08567758e-01],
```

```
[ 5.85442370e-01],
[ 7.96154407e-01],
[ 3.03635607e+00],
[-5.67928784e-01],
[-2.57405781e-01],
[ 1.49483116e+00],
[-1.02144280e-01],
[-1.31096597e+00],
[ 5.31172216e-02],
[ 5.31172216e-02],
[-2.79585996e-01],
[ 4.20271144e-02],
[-8.89541895e-01],
[-6.67739750e-01],
[ 1.17321805e+00],
[-4.12667283e-01],
[-1.18897479e+00],
[ 9.07055480e-01],
[ 3.03635607e+00],
[ 2.63829259e-01],
[ 2.19468830e-01],
[-1.25551543e+00],
[ 2.84782425e+00],
[-8.45181466e-01],
[ 1.41838080e-01],
[-7.99640654e-02],
[ 8.29424729e-01],
[-9.78262753e-01],
[-1.37750661e+00],
[ 9.74776506e-02],
[-4.68117819e-01],
[-1.57594816e-01],
[-1.13234387e-01],
[ 3.03635607e+00],
[ 1.04013677e+00],
[-5.12478248e-01],
[-1.05589350e+00],
[-1.16679458e+00],
[-8.45181466e-01],
[ 6.40892906e-01],
[-2.13045352e-01],
[ 2.41649045e-01],
[-1.04480340e+00],
[ 5.85442370e-01],
[-5.90108999e-01],
[ 4.20271144e-02],
[-1.02144280e-01],
[-7.01010071e-01],
[-3.01766210e-01],
[-4.79207926e-01],
```

```
[-4.79207926e-01],
[ 4.41270975e-01],
[ 1.64018294e-01],
[ 8.75679261e-03],
[ 2.63829259e-01],
[-5.34658463e-01],
[-6.01199106e-01],
[ 3.03635607e+00],
[ 2.19468830e-01],
[-1.79775031e-01],
[-1.35532640e+00],
[-1.57712854e+00],
[ 1.48374106e+00],
[-4.34847497e-01],
[ 1.97288616e-01],
[-4.23757390e-01],
[ 1.86198509e-01],
[ 2.08378723e-01],
[ 2.97099581e-01],
[ 3.09370071e-02],
[-1.68802962e+00],
[-2.24135460e-01],
[-1.95419219e+00],
[-1.57712854e+00],
[-5.45748570e-01],
[-3.01766210e-01],
[-1.21115500e+00],
[-4.66937436e-02],
[ 1.61682234e+00],
[-4.66937436e-02],
[ 7.18523657e-01],
[-8.23001251e-01],
[ 4.30180868e-01],
[-4.12667283e-01],
[-9.67172645e-01],
[-2.24135460e-01],
[ 7.18523657e-01],
[ 1.41720041e+00],
[ 1.41838080e-01],
[ 6.42073289e-02],
[ 2.41649045e-01],
[-1.05589350e+00],
[-5.34658463e-01],
[-1.20006490e+00],
[ 1.64018294e-01],
[ 2.45967049e+00],
[-1.03371329e+00],
[-1.24324494e-01],
[-1.05589350e+00],
[-9.78262753e-01],
```

```
[-5.23568355e-01],
[-3.79396961e-01],
[-1.02262318e+00],
[-8.56271573e-01],
[-1.35414602e-01],
[-1.02144280e-01],
[ 1.06231698e+00],
[-7.78640822e-01],
[ 1.30747972e-01],
[-5.34658463e-01],
[-1.43295715e+00],
[-1.00044297e+00],
[-9.78262753e-01],
[-2.01955245e-01],
[ 2.63829259e-01],
[-1.01153307e+00],
[-1.20006490e+00],
[-2.46315674e-01],
[ 1.69445309e+00],
[ 2.11587717e+00],
[ 8.63875433e-02],
[-1.23333522e+00],
[ 9.95776338e-01],
[-3.35036532e-01],
[-1.57594816e-01],
[ 3.09370071e-02],
[ 4.41270975e-01],
[-7.78640822e-01],
[ 1.13994773e+00],
[ 1.53919159e+00],
[-1.03371329e+00],
[ 1.17321805e+00],
[ 8.29424729e-01],
[ 4.52361083e-01],
[-6.88739581e-02],
[-3.57216747e-01],
[-6.78829857e-01],
[-1.02144280e-01],
[ 5.31172216e-02],
[ 1.62791245e+00],
[ 6.74163228e-01],
[-6.88739581e-02],
[-9.22812216e-01],
[ 1.19657865e-01],
[-9.10541726e-02],
[-2.79585996e-01],
[ 6.18712691e-01],
[ 5.41081941e-01],
[-2.46315674e-01],
[ 6.63073120e-01],
```

```
[ 3.03635607e+00],
[-1.57594816e-01],
[-2.35225567e-01],
[ 8.75679261e-03],
[ 2.63829259e-01],
[ 2.90327478e+00],
[ 1.16212795e+00],
[-4.12667283e-01],
[ 2.23786835e+00],
[ 1.19539827e+00],
[ 1.18430816e+00],
[-2.57405781e-01],
[ 2.63829259e-01],
[-8.56271573e-01],
[ 1.97288616e-01],
[ 2.37094964e+00],
[-4.45937605e-01],
[ 1.30747972e-01],
[ 2.41649045e-01],
[-2.90676103e-01],
[-3.68306854e-01],
[ 9.84686231e-01],
[-4.57027712e-01],
[ 6.40892906e-01],
[-4.34847497e-01],
[-5.79018892e-01],
[-2.79585996e-01],
[-2.24135460e-01],
[ 1.19657865e-01],
[ 2.30558938e-01],
[-4.66937436e-02],
[ 2.27113867e+00],
[-2.90676103e-01],
[ 7.07433549e-01],
[ 1.90516513e+00],
[ 6.42073289e-02],
[-3.01766210e-01],
[-1.67693951e+00],
[ 2.63829259e-01],
[-1.57594816e-01],
[-1.34234219e-02],
[-2.45135291e-02],
[-1.02262318e+00],
[ 1.75108401e-01],
[-3.35036532e-01],
[-2.01955245e-01],
[ 7.73974193e-01],
[-3.46126639e-01],
[ 1.52928187e-01],
[-2.13045352e-01],
```

```
[-1.06698361e+00],
[ 5.31172216e-02],
[-5.79018892e-01],
[ 2.08378723e-01],
[-5.67928784e-01],
[-1.10025393e+00],
[-3.46126639e-01],
[-2.46315674e-01],
[ 1.98468999e-02],
[ 1.39502020e+00],
[ 2.18241781e+00],
[-1.53276812e+00],
[ 1.52810149e+00],
[ 1.08449720e+00],
[ 5.31172216e-02],
[-1.02144280e-01],
[ 5.41081941e-01],
[ 1.30747972e-01],
[ 1.16212795e+00],
[-5.67928784e-01],
[-1.73239005e+00],
[ 3.03635607e+00],
[ 1.19657865e-01],
[-3.57216747e-01],
[ 3.03635607e+00],
[ 1.50592127e+00],
[ 2.63829259e-01],
[-1.58821865e+00],
[-3.46126639e-01],
[-1.03371329e+00],
[ 1.33956966e+00],
[-3.46126639e-01],
[ 2.41649045e-01],
[-9.44992431e-01],
[ 1.09558730e+00],
[-1.68684923e-01],
[-3.12856318e-01],
[ 9.40325802e-01],
[-9.44992431e-01],
[-3.57216747e-01],
[-5.01388141e-01],
[ 1.41838080e-01],
[-8.00821037e-01],
[-6.88739581e-02],
[ 1.50592127e+00],
[ 5.41081941e-01],
[-3.68306854e-01],
[-4.57027712e-01],
[-9.11722109e-01],
[-1.71020983e+00],
```

```
       [-1.71020983e+00],
       [-1.37750661e+00],
       [-7.99640654e-02],
       [-7.78640822e-01],
       [-1.34423629e+00],
       [-1.34234219e-02],
       [ 2.63829259e-01],
       [-6.12289213e-01],
       [-1.88765155e+00],
       [ 7.52974361e-02],
       [-5.34658463e-01],
       [-6.45559535e-01],
       [ 2.67038253e+00],
       [-7.78640822e-01],
       [-6.88739581e-02],
       [-6.88739581e-02],
       [ 6.51983013e-01],
       [-4.90298034e-01],
       [ 1.66118277e+00],
       [-3.57216747e-01],
       [ 8.63875433e-02],
       [-4.90298034e-01],
       [-1.44404726e+00],
       [ 3.09370071e-02],
       [-1.10025393e+00],
       [-7.23190286e-01],
       [-2.57405781e-01],
       [ 1.41838080e-01],
       [-2.33331464e-03],
       [-4.57027712e-01],
       [ 2.52621114e+00],
       [-6.78829857e-01]])
```

```
X_test
```

```
array([[-4.25535687e-01, -1.38516090e+00, -1.28660800e+00,
         1.41766060e+00, -2.78981450e+00, -8.22034273e-01],
       [-3.72998223e-01, -7.01948458e-01, -4.22878815e-01,
         3.92416981e-01, -5.13737140e-01, -1.21532436e+00],
       [-4.24548903e-01,  1.34548824e-01,  1.73094321e-01,
        -2.21025187e-01,  1.19332088e+00, -4.83749797e-01],
       [-2.54305939e-01, -4.18736464e-01, -1.29210893e-01,
        -2.89185428e-01,  1.19332088e+00,  6.08057020e-02],
       [-3.97468902e-01, -1.61801871e-01, -7.73871416e-02,
        -8.10327270e-01, -3.95543560e-02, -1.50965881e-01],
       [-1.11968547e-01,  1.25133839e+00,  4.49487660e-01,
         1.79416228e-01, -1.79403066e+00, -1.75718403e-01],
       [-4.17324189e-01, -1.45743459e-01, -5.14752661e-02,
         6.23877799e-01, -3.24064026e-01, -3.35234661e-01],
       [-4.00572266e-01, -6.10718321e-02, -5.52438192e-01,
         7.14958467e-02,  5.52822008e-02, -2.23848309e-01],
```

```
[-4.13790489e-01, -3.57422527e-01, -2.84682145e-01,
  2.81656590e-01,  1.14590260e+00, -4.50800894e-02],
[ 1.01104996e+00,  1.03527975e+00,  1.42550164e+00,
  1.92196273e-01,  8.13974655e-01,  3.33083451e-01],
[-4.25401775e-01, -8.30415754e-01, -1.27797071e+00,
 -9.46447404e-02, -8.93083367e-01, -7.50526985e-01],
[ 4.74721896e-01,  1.03527975e+00,  1.27003038e+00,
 -7.98967230e-01,  8.13974655e-01,  6.20487742e-01],
[-3.00104930e-01, -4.18736464e-01, -1.29210893e-01,
  5.65657593e-01,  1.19332088e+00, -8.49591537e-02],
[-4.28666352e-01, -1.02311670e+00, -1.23478425e+00,
 -5.50466351e-01,  1.24073916e+00,  3.02830368e-01],
[-4.25871115e-01, -1.21581764e+00, -1.18209677e+00,
  2.23984351e+00, -1.79403066e+00, -1.20844865e+00],
[-4.20098625e-01, -1.36034335e+00, -1.22614695e+00,
  4.30757116e-01, -8.69726343e-02, -9.77425110e-01],
[-4.22027989e-01, -1.28005129e+00, -1.32115716e+00,
  9.84759421e-02, -1.74661238e+00, -1.04893240e+00],
[-8.77929930e-02, -1.61801871e-01, -7.73871416e-02,
 -1.84977094e+00, -3.95543560e-02,  5.80009607e-03],
[-4.24368187e-01, -1.38516090e+00, -1.28660800e+00,
  1.18051977e+00, -2.78981450e+00, -5.48381384e-01],
[-2.32945125e-01,  1.25133839e+00,  4.49487660e-01,
  2.98108613e+00, -1.79403066e+00, -1.27583052e+00],
[-3.60214237e-01, -1.02749627e+00,  8.12253916e-01,
  1.31684025e+00, -2.60014139e+00, -4.13617649e-01],
[ 1.11374543e+01,  1.03527975e+00,  1.01954892e+00,
  9.83139069e-01,  8.13974655e-01,  6.34239144e-01],
[-4.13457660e-01, -1.45743459e-01, -5.14752661e-02,
 -5.73186432e-01, -3.24064026e-01,  3.81213356e-01],
[ 4.82579764e-01,  1.03527975e+00,  1.42550164e+00,
 -3.82905759e-01,  8.13974655e-01,  4.26592981e-01],
[ 2.43645206e+00,  1.03527975e+00,  1.61552206e+00,
 -6.49866703e-01,  8.13974655e-01,  1.30805782e+00],
[ 1.76564966e-01,  1.03527975e+00,  1.38231518e+00,
 -4.25505910e-01,  8.13974655e-01,  8.81764370e-01],
[-4.28483037e-01, -1.31362797e+00, -1.01885195e+00,
  5.10277398e-01, -7.03410253e-01, -9.08668102e-01],
[ 2.02902604e-01,  1.03527975e+00,  5.27223286e-01,
  2.95856640e-01,  8.13974655e-01,  8.36384745e-01],
[ 1.26947292e+00,  1.03527975e+00,  2.33555364e-01,
 -7.98967230e-01,  8.13974655e-01,  2.97329807e-01],
[-4.23904048e-01, -7.22386437e-01, -9.93803804e-01,
 -8.08907265e-01, -7.50828531e-01, -2.30778470e-02],
[-3.87554260e-01,  1.58856504e+00,  6.13596204e-01,
 -6.44186683e-01,  1.28815744e+00,  3.34458591e-01],
[-3.60618571e-01, -7.01948458e-01, -4.22878815e-01,
 -4.18405885e-01, -5.13737140e-01, -1.30338779e-01],
[-3.61927783e-01, -7.01948458e-01, -4.22878815e-01,
  3.47808789e+00, -5.13737140e-01, -1.09568716e+00],
```

```
[-4.08644652e-01, -5.29685493e-01, -5.17889025e-01,
  2.13496349e-01,  5.29464984e-01, -7.42276144e-01],
[-3.28284824e-01, -1.02749627e+00,  8.12253916e-01,
  1.04845930e+00, -2.60014139e+00,  3.01455228e-01],
[-4.26226045e-01, -1.28881043e+00, -8.20194238e-01,
  1.02573922e+00,  1.02700479e-01, -1.32808585e+00],
[ 6.21778431e+00,  1.03527975e+00,  3.80389325e-01,
 -7.36487009e-01,  8.13974655e-01, -3.42110361e-01],
[ 4.46602848e+00,  1.03527975e+00,  1.08864725e+00,
 -1.04584776e-01,  8.13974655e-01,  2.64326444e-01],
[ 1.44510743e-01,  1.03527975e+00,  2.68104531e-01,
 -3.87165774e-01,  8.13974655e-01,  1.19942174e+00],
[-4.20015418e-01, -7.36984993e-01, -4.66065274e-01,
 -6.04426542e-01,  3.39791871e-01, -5.26379141e-01],
[-4.18955828e-01, -4.58152566e-01, -2.50132978e-01,
 -3.74385729e-01, -1.55693926e+00, -2.30778470e-02],
[-4.22291911e-01,  1.34548824e-01,  1.73094321e-01,
  4.50637187e-01,  1.19332088e+00, -4.02616528e-01],
[ 6.03545931e-02,  1.03527975e+00,  8.81352251e-01,
 -4.59586030e-01,  8.13974655e-01,  6.25988303e-01],
[-4.21831672e-01, -1.13406573e+00, -8.02919655e-01,
 -5.78866452e-01, -7.50828531e-01, -3.61362323e-01],
[-2.64870637e-01,  1.25133839e+00,  4.49487660e-01,
 -3.64245347e-02, -1.79403066e+00, -9.76049970e-01],
[-4.23450309e-01,  4.25060096e-01, -1.00157737e+00,
  3.88156966e-01, -1.17759304e+00, -7.16148482e-01],
[ 3.09555864e-01,  1.03527975e+00,  2.59467239e-01,
 -2.29545217e-01,  8.13974655e-01,  3.27582891e-01],
[-7.49882052e-02,  1.25133839e+00,  2.74700728e+00,
 -9.63687812e-01, -1.79403066e+00,  1.21729857e+00],
[ 1.59457854e+00,  1.03527975e+00,  2.33555364e-01,
 -4.96506161e-01,  8.13974655e-01,  7.60752037e-01],
[ 6.35731475e-01,  1.03527975e+00,  1.38231518e+00,
 -4.82306111e-01,  8.13974655e-01,  5.97110360e-01],
[-4.25821711e-01, -1.07567150e+00, -1.39025550e+00,
 -5.67506412e-01, -9.87919923e-01, -4.60372414e-01],
[-4.02922865e-01, -3.57422527e-01, -2.84682145e-01,
 -1.96885102e-01,  1.14590260e+00,  1.15811308e-01],
[-4.24795924e-01, -7.83700374e-01, -9.84302783e-01,
 -2.28125212e-01, -7.98246810e-01, -4.35619891e-01],
[-4.05637496e-01, -7.51583550e-01, -1.05340112e+00,
  6.28137814e-01,  2.92373593e-01, -8.30285114e-01],
[ 1.52277344e+00,  1.03527975e+00,  5.27223286e-01,
 -1.37974928e+00,  8.13974655e-01,  1.69309706e+00],
[-3.92783304e-01, -1.92458840e-01,  2.76741823e-01,
 -1.25762885e+00,  3.39791871e-01,  1.17466922e+00],
[-4.19746295e-01, -7.51583550e-01, -1.05340112e+00,
  9.67519014e-01,  2.92373593e-01, -1.24695258e+00],
[-4.26067432e-01, -1.17640154e+00, -1.07931299e+00,
  1.06265935e+00, -8.69726343e-02, -1.46009930e+00],
```

```
[ 1.63157968e+00,  1.03527975e+00,  1.08864725e+00,
 -5.39106311e-01,  8.13974655e-01,  1.62158977e+00],
[-4.23094079e-01,  2.65935831e-01, -1.00157737e+00,
 -6.19846250e-02,  1.02700479e-01, -3.54541084e-02],
[ 6.73636196e-01,  1.03527975e+00,  2.68104531e-01,
  1.02735957e-01,  8.13974655e-01,  6.93370170e-01],
[ 1.44670397e+00,  1.03527975e+00,  3.80389325e-01,
  8.18418487e-01,  8.13974655e-01,  9.87650162e-01],
[-1.51017364e-01,  1.25133839e+00,  2.74700728e+00,
 -8.04647250e-01, -1.79403066e+00,  4.38969243e-01],
[-4.28743059e-01, -1.38516090e+00, -9.58390908e-01,
  1.37080044e+00, -1.41468443e+00, -9.77425110e-01],
[-4.25284765e-01, -8.49393878e-01, -3.27868604e-01,
 -3.38885604e-01,  8.13974655e-01, -6.30889792e-01],
[-4.12165351e-01, -6.10718321e-02, -5.52438192e-01,
  1.41076093e-01,  5.52822008e-02, -4.42495592e-01],
[-3.88487739e-01,  1.58856504e+00,  6.13596204e-01,
 -4.73786080e-01,  1.28815744e+00,  5.91609799e-01],
[-4.28212614e-01, -1.31216812e+00, -1.23478425e+00,
  6.42337864e-01, -7.03410253e-01, -1.11356398e+00],
[-4.08460036e-01,  2.13747076e+00,  2.42192656e-01,
 -4.11305860e-01,  2.92373593e-01,  3.04205508e-01],
[-4.13772287e-01, -7.20926581e-01, -1.24342154e+00,
 -5.54726366e-01,  1.97537036e-01, -5.23628861e-01],
[ 1.63341284e+00,  1.03527975e+00,  1.01954892e+00,
  3.82476946e-01,  8.13974655e-01,  1.16641838e+00],
[-3.77427699e-01, -7.01948458e-01, -3.96966939e-01,
 -1.58544966e-01, -5.13737140e-01,  1.21867371e+00],
[ 2.18082434e+00,  1.03527975e+00,  1.27003038e+00,
 -2.70887398e+00,  8.13974655e-01,  2.47967722e+00],
[-2.70199790e-01, -4.18736464e-01, -1.29210893e-01,
 -1.89785077e-01,  1.19332088e+00,  8.41885306e-01],
[-4.13865895e-01, -5.98298708e-01, -9.06567157e-01,
  7.01978075e-01, -2.76645748e-01, -1.06680922e+00],
[ 1.54276913e+00,  1.03527975e+00,  1.61552206e+00,
 -1.75585027e-01,  8.13974655e-01,  1.90486864e+00],
[-4.09715943e-01, -5.98298708e-01, -9.06567157e-01,
 -9.18047304e-02, -2.76645748e-01, -7.09272781e-01],
[-4.14725268e-01, -1.18516067e+00, -9.32479032e-01,
  2.54656459e+00, -2.29227469e-01, -1.15344305e+00],
[ 8.58826529e-01,  1.03527975e+00,  1.20956934e+00,
 -6.01586532e-01,  8.13974655e-01,  2.38891797e+00],
[-4.24344785e-01, -1.10924818e+00, -5.52438192e-01,
  2.00716303e-01, -3.24064026e-01, -5.20878581e-01],
[-4.19790498e-01, -1.18516067e+00, -9.32479032e-01,
  2.19298334e+00, -2.29227469e-01, -1.24145202e+00],
[-3.55243915e-01, -1.02749627e+00,  1.90368905e-01,
  2.87032574e+00, -2.60014139e+00, -7.09272781e-01],
[-3.01927685e-01, -4.18736464e-01, -1.29210893e-01,
 -6.56966728e-01,  1.19332088e+00,  1.00140156e+00],
```

```
       [ 1.37846118e+00,  1.03527975e+00,  1.38231518e+00,
        -9.60647455e-02,  8.13974655e-01,  3.53710554e-01],
       [ 3.32996856e-01,  1.03527975e+00,  1.20956934e+00,
         1.83676243e-01,  8.13974655e-01,  9.31269416e-01],
       [-2.67687976e-01, -4.18736464e-01, -1.29210893e-01,
        -1.00202795e+00,  1.19332088e+00,  1.15816754e+00],
       [-2.15058205e-01,  1.25133839e+00,  2.74700728e+00,
        -2.18185177e-01, -1.79403066e+00,  2.06570558e-01],
       [-1.29601945e-01,  1.25133839e+00,  4.49487660e-01,
        -5.61826392e-01, -1.79403066e+00, -7.80834530e-02],
       [ 7.83724348e-01,  1.03527975e+00,  1.08864725e+00,
         1.48176118e-01,  8.13974655e-01,  1.57896043e+00],
       [-4.22423222e-01, -9.66182329e-01, -9.58390908e-01,
        -5.36266301e-01,  1.50118758e-01,  9.92551651e-03],
       [-4.14080413e-01, -1.10486861e+00, -1.00157737e+00,
         3.98097001e-01, -1.55693926e+00, -1.10531314e+00],
       [-4.12698396e-01, -6.10718321e-02, -5.52438192e-01,
        -5.46206336e-01,  5.52822008e-02, -2.37599710e-01],
       [-4.26883901e-01, -5.74941017e-01, -7.25184028e-01,
         2.06396324e-01, -3.24064026e-01, -4.75498956e-01],
       [-4.16098185e-01, -8.58153012e-01, -8.63380698e-01,
         2.56096499e-01,  5.76883263e-01, -8.06907731e-01],
       [-4.11641406e-01, -4.58152566e-01, -2.50132978e-01,
        -1.47184926e-01, -1.55693926e+00,  9.01016332e-01],
       [-2.46618386e-01,  1.25133839e+00,  2.74700728e+00,
        -2.08245142e-01, -1.79403066e+00,  3.46834853e-01],
       [-4.21465041e-01,  2.13747076e+00,  2.42192656e-01,
        -5.76026442e-01,  2.92373593e-01,  2.43699342e-01],
       [-1.69087600e-01,  1.25133839e+00,  4.49487660e-01,
         2.34776389e+00, -1.79403066e+00, -1.22357520e+00],
       [ 1.42042874e+00,  1.03527975e+00,  1.20956934e+00,
         9.56359321e-02,  8.13974655e-01,  1.06190773e+00],
       [ 2.06388200e-01,  1.03527975e+00,  6.74057247e-01,
        -1.85403096e+00,  8.13974655e-01, -1.28408136e+00],
       [-3.92614290e-01,  1.58856504e+00,  6.13596204e-01,
        -1.44344916e-01,  1.28815744e+00,  1.58996155e+00],
       [ 5.27371169e-01,  1.03527975e+00,  1.08864725e+00,
        -1.17364821e-01,  8.13974655e-01,  1.22692455e+00]])
```

```
Y_test
```

```
array([[ 1.32847956e+00],
       [ 9.84686231e-01],
       [-2.24135460e-01],
       [-9.00632002e-01],
       [-7.12100179e-01],
       [-3.56036364e-02],
       [ 1.98468999e-02],
       [ 1.97288616e-01],
       [-3.12856318e-01],
       [-6.56649642e-01],
```

```
[ 6.42073289e-02],
[-8.34091358e-01],
[-1.79775031e-01],
[-4.12667283e-01],
[ 2.87000446e+00],
[ 1.64018294e-01],
[-1.34234219e-02],
[-7.23190286e-01],
[ 8.51604944e-01],
[ 3.03635607e+00],
[ 1.23975870e+00],
[-1.35532640e+00],
[-2.46315674e-01],
[-9.33902324e-01],
[-1.34423629e+00],
[-1.10025393e+00],
[ 2.08378723e-01],
[-6.56649642e-01],
[-2.79585996e-01],
[-6.12289213e-01],
[-4.68117819e-01],
[ 1.86198509e-01],
[ 3.03635607e+00],
[ 1.30747972e-01],
[ 8.95965373e-01],
[ 1.19539827e+00],
[-8.45181466e-01],
[-1.29987586e+00],
[-3.90487068e-01],
[-1.79775031e-01],
[ 3.09370071e-02],
[-2.45135291e-02],
[-4.01577176e-01],
[-3.68306854e-01],
[ 4.85631404e-01],
[ 4.96721512e-01],
[-3.90487068e-01],
[-8.00821037e-01],
[-3.90487068e-01],
[-1.01153307e+00],
[-1.90865138e-01],
[-3.68306854e-01],
[-2.35225567e-01],
[ 3.96910546e-01],
[-1.17788468e+00],
[-3.23946425e-01],
[ 7.73974193e-01],
[ 1.36174988e+00],
[-1.58821865e+00],
[-1.57594816e-01],
```

```
       [-9.00632002e-01],
       [ 5.41081941e-01],
       [-3.57216747e-01],
       [ 1.11776752e+00],
       [-1.68684923e-01],
       [ 6.07622584e-01],
       [-5.79018892e-01],
       [ 8.29424729e-01],
       [-1.35414602e-01],
       [-1.90865138e-01],
       [-1.29987586e+00],
       [-1.02144280e-01],
       [-1.53276812e+00],
       [-8.23001251e-01],
       [ 4.41270975e-01],
       [-1.54385822e+00],
       [ 2.30558938e-01],
       [ 1.78317395e+00],
       [-1.81002080e+00],
       [-2.33331464e-03],
       [ 2.34876942e+00],
       [ 3.03635607e+00],
       [-9.00632002e-01],
       [-1.21115500e+00],
       [-1.12243415e+00],
       [-1.00044297e+00],
       [-1.24324494e-01],
       [-3.90487068e-01],
       [-1.45513736e+00],
       [-6.01199106e-01],
       [ 7.96154407e-01],
       [-2.33331464e-03],
       [-1.13234387e-01],
       [-4.66937436e-02],
       [ 4.96721512e-01],
       [-6.23379321e-01],
       [-6.88739581e-02],
       [ 3.03635607e+00],
       [-1.71020983e+00],
       [ 3.03635607e+00],
       [-9.56082538e-01],
       [-1.28878576e+00]])
```

Multiple Regression Model

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X_train, Y_train)#training func question + answers
# The coefficients
```

```
print ('Intercept: ',regr.intercept_)
print ('Coefficient : ',regr.coef_)
```

```
Intercept:  [4.26749913e-15]
Coefficient :  [[-0.0208389   0.02191235 -0.04117471  0.34457457 -
0.22306074 -0.41782379]]
```

```
regr.intercept_
```

```
array([4.26749913e-15])
```

```
regr.coef_
```

```
array([[-0.0208389 ,  0.02191235, -0.04117471,  0.34457457, -
0.22306074,
        -0.41782379]])
```

```
regr.coef_[0]
```

```
array([-0.0208389 ,  0.02191235, -0.04117471,  0.34457457, -
0.22306074,
       -0.41782379])
```

```
regr.coef_[0][1]
```

```
0.021912354709027443
```

```
regr.coef_[0][2]
```

```
-0.041174713622634834
```

```
y_pred = regr.predict(X_test)
```

```
y_pred
```

```
array([[ 1.48574464],
       [ 0.76740637],
       [-0.13555222],
       [-0.38979088],
       [-0.19939426],
       [ 0.54666495],
       [ 0.43494988],
       [ 0.13558925],
       [-0.12720597],
       [-0.31158841],
       [ 0.52347632],
       [-0.75562453],
       [-0.03337528],
       [-0.55561071],
       [ 1.70779528],
       [ 0.60565191],
       [ 0.8969458 ],
       [-0.62951399],
```

```
[ 1.28966892],
[ 1.9742232 ],
[ 1.15810571],
[-0.35918783],
[-0.27695762],
[-0.53781171],
[-1.04663724],
[-0.73451687],
[ 0.73449035],
[-0.43233368],
[-0.59448663],
[-0.06768194],
[-0.6314317 ],
[ 0.03442667],
[ 1.78043215],
[ 0.28383618],
[ 0.76618863],
[ 0.89985355],
[-0.41494745],
[-0.44325138],
[-0.80748537],
[-0.05233683],
[ 0.23692123],
[ 0.06193911],
[-0.61634199],
[ 0.13600393],
[ 0.80987559],
[ 0.75502564],
[-0.39198204],
[-0.52462561],
[-0.69066991],
[-0.64472209],
[ 0.25971895],
[-0.35955   ],
[ 0.31367168],
[ 0.53349369],
[-1.39516432],
[-1.00737272],
[ 0.82482341],
[ 1.02317125],
[-1.10100655],
[ 0.02643056],
[-0.43826369],
[-0.33534805],
[-0.1430354 ],
[ 1.21433887],
[-0.03098624],
[ 0.25116246],
[-0.68014043],
[ 0.87452271],
```

```
       [-0.28867025],
       [ 0.02760012],
       [-0.59046406],
       [-0.44039801],
       [-2.22609673],
       [-0.68156247],
       [ 0.78217289],
       [-1.11395022],
       [ 0.35918181],
       [ 1.43161632],
       [-1.43201923],
       [ 0.36636628],
       [ 1.34665901],
       [ 1.84243142],
       [-0.9085299 ],
       [-0.42541258],
       [-0.54143969],
       [-1.0936433 ],
       [ 0.15748076],
       [ 0.25082485],
       [-0.82870657],
       [-0.19532347],
       [ 0.97195063],
       [-0.07125685],
       [ 0.36823639],
       [ 0.32212591],
       [-0.07105211],
       [ 0.10295776],
       [-0.31987684],
       [ 1.73283222],
       [-0.64902082],
       [-0.29326758],
       [-0.98367249],
       [-0.76777415]])
```

```
from sklearn.metrics import r2_score
```

```
print(f"R2 Score : {r2_score(Y_test,y_pred)*100} % ")
```

```
R2 Score : 70.70768805651036 %
```

```
print(f"Mean absolute error: {np.mean(np.absolute(y_pred -
Y_test))*100}% ")#pred - actual
```

```
Mean absolute error: 41.626666556088765%
```

```
print("Residual sum of squares (MSE): %.2f" % np.mean((y_pred -
Y_test) **2))
```

```
Residual sum of squares (MSE): 0.35
```

```
accuracy = (f"accuracy : {r2_score(Y_test,y_pred)*100} % ")
```

```
accuracy
```

```
'accuracy : 70.70768805651036 % '
```

Apply Lasso Regression to cover the overfitting and underfitting problem

```
from sklearn.linear_model import Lasso
```

```
L = Lasso(alpha = 1)
```

```
L.fit(X_train,Y_train)
```

```
Lasso(alpha=1)
```

```
ypred1 = L.predict(X_test)
```

```
from sklearn.metrics import r2_score
print("R2-score",r2_score(Y_test,ypred1))
print(f"Mean absolute error: {np.mean(np.absolute(ypred1 - Y_test))}
")# pred - actual
```

```
R2-score -0.001983416409810257
Mean absolute error: 0.7986761095147675
```

Ans: The overfitting and underfitting problem is not there in the problem because the lasso gives the less accuracy

Ans : The best accuracy of the model will be 70.70%