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

In [1]:

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

In [2]:

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

In [3]:

```
diabetes = load_diabetes()
```

In [4]:

```
diabetes
```

Out[4]:

```
{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
          0.01990842, -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., 31
0., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
```

```
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTe
n baseline variables, age, sex, body mass index, average blood\npressure, an
d six blood serum measurements were obtained for each of n =\n442 diabetes p
atients, as well as the response of interest, a\nquantitative measure of dis
ease progression one year after baseline.\n\n**Data Set Characteristics:**\n
\n  :Number of Instances: 442\n\n  :Number of Attributes: First 10 columns a
re numeric predictive values\n\n  :Target: Column 11 is a quantitative measu
re of disease progression one year after baseline\n\n  :Attribute Informatio
n:\n      - age     age in years\n      - sex\n      - bmi     body mass ind
ex\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. t
he sum of squares of each column totals 1).\n\nSource URL:\nhttps://www4.sta
t.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBrad
ley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.\n(http
s://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.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'}
```


In [5]:

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

Out[5]:

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

In [6]:

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

Out[6]:

```
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]])
```

In [7]:

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

Out[7]:

```
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.])
```

In [8]:

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

Out[8]:

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

In [9]:

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

Out[9]:

```
'.. _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 progres
sion 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 p
redictive values\n\n  :Target: Column 11 is a quantitative measure of diseas
e 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-d
ensity lipoproteins\n      - s4      tch, total cholesterol / HDL\n      - s
5      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 su
m of squares of each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncs
u.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley E
fron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angl
e Regression," Annals of Statistics (with discussion), 407-499.\n(https://we
b.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)'
```

In [10]:

```
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 ye
ar 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 (https://www4.sta
t.ncsu.edu/~boos/var.select/diabetes.html)

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

In [11]:

```
diabetes
```

Out[11]:


```
{'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., 31
0., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
```

```
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTe
n baseline variables, age, sex, body mass index, average blood\npressure, an
d six blood serum measurements were obtained for each of n =\n442 diabetes p
atients, as well as the response of interest, a\nquantitative measure of dis
ease progression one year after baseline.\n\n**Data Set Characteristics:**\n
\n  :Number of Instances: 442\n\n  :Number of Attributes: First 10 columns a
re numeric predictive values\n\n  :Target: Column 11 is a quantitative measu
re of disease progression one year after baseline\n\n  :Attribute Informatio
n:\n      - age     age in years\n      - sex\n      - bmi     body mass ind
ex\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. t
he sum of squares of each column totals 1).\n\nSource URL:\nhttps://www4.sta
t.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBrad
ley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.\n(http
s://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.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'}
```


In [12]:

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

In [13]:

```
df
```

Out[13]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 10 columns

In [14]:

```
diabetes["target"]
```

Out[14]:

```
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.])
```

In [15]:

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

In [16]:

```
df
```

Out[16]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 11 columns

In [17]:

```
df.head()
```

Out[17]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031

In [18]:

```
df.tail()
```

Out[18]:

	age	sex	bmi	bp	s1	s2	s3	s4	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

In [19]:

```
df.shape
```

Out[19]:

```
(442, 11)
```

In [20]:

```
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
```

In [21]:

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

Out[21]:

```
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
```

In [22]:

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

In [23]:

```
scaler = StandardScaler()
```

In [24]:

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

Out[24]:

```
StandardScaler()
```

In [25]:

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

In [26]:

```
scaled_data
```

Out[26]:

```
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]])
```

In [27]:

```
scaled_data.shape
```

Out[27]:

```
(442, 11)
```

In [28]:

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

Out[28]:

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

In [29]:

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

Out[29]:

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

In [30]:

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

Out[30]:

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

In [31]:

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

In [32]:

```
columns_having_var_more_than_50
```

Out[32]:

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

In [33]:

```
len(columns_having_var_more_than_50)
```

Out[33]:

```
11
```

In [34]:

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

Out[34]:

```
11
```

In [35]:

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

Out[35]:

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

In [36]:

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

In [37]:

```
columns_having_var_less_than_50
```

Out[37]:

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

In [38]:

```
len(columns_having_var_less_than_50)
```

Out[38]:

```
0
```

In [40]:

```
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

In [41]:

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

In [42]:

```
cdf
```

Out[42]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 10 columns

In [43]:

```
cdf.head()
```

Out[43]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031

In [44]:

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

In [45]:

```
cdf
```

Out[45]:

	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 × 1 columns

In [46]:

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

In [47]:

```
X
```

Out[47]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 10 columns

In [48]:

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

In [49]:

```
Y
```

Out[49]:

	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 × 1 columns

In [50]:

```
# 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
```

In [51]:

```
X_train.shape
```

Out[51]:

```
(353, 10)
```

In [52]:

```
X_test.shape
```

Out[52]:

```
(89, 10)
```

In [53]:

```
Y_train.shape
```

Out[53]:

```
(353, 1)
```

In [54]:

```
Y_test.shape
```

Out[54]:

```
(89, 1)
```

In [55]:

```
# 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)
```

In [56]:

```
Y_train
```

Out[56]:

```
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],
```

In [57]:

```
Y_test
```

Out[57]:

```
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

In [58]:

```
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]]
```

In [59]:

```
regr.intercept_
```

Out[59]:

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

In [60]:

```
regr.coef_
```

Out[60]:

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

In [61]:

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

Out[61]:

```
-0.45411742742487105
```

In [62]:

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

Out[62]:

```
3.901364270938186
```

In [63]:

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

In [64]:

```
y_pred
```

Out[64]:

```
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]])
```

In [65]:

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

In [66]:

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

```
R2 Score : 33.222203269065155 %
```

In [67]:

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

```
Mean absolute error: 0.589718094672523
```

In [68]:

```
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

In [69]:

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

In [70]:

```
data = load_wine()
```

In [71]:

```
data
```

Out[71]:


```
{'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, 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- Proant
hocyanins\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    Mal
ic 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    1
9.5   3.3\n    Magnesium:                  70.0 162.0    99.7  14.3\n    T
otal 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    Co
lour 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 Attribu
te Values: None\n    :Class Distribution: class_0 (59), class_1 (71), class_
2 (48)\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PL
U@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThis is a copy of UCI ML Wine r
ecognition datasets.\nhttps://archive.ics.uci.edu/ml/machine-learning-databa
ses/wine/wine.data\n\nThe data is the results of a chemical analysis of wine
s grown in the same\nregion in Italy by three different cultivators. There a
re thirteen different\nmeasurements taken for different constituents found i
n the three types of\nwine.\n\nOriginal Owners: \n\nForina, M. et al, PARVUS
```

```
- \nAn Extendible Package for Data Exploration, Classification and Correlati
on. \nInstitute of Pharmaceutical and Food Analysis and Technologies,\nVia B
rigata Salerno, 16147 Genoa, Italy.\n\nCitation:\n\nLichman, M. (2013). UCI
Machine Learning Repository\n[https://archive.ics.uci.edu/ml]. Irvine, CA: U
niversity of California,\nSchool of Information and Computer Science. \n\n..
topic:: References\n\n  (1) S. Aeberhard, D. Coomans and O. de Vel, \n  Comp
arison of Classifiers in High Dimensional Settings, \n  Tech. Rep. no. 92-0
2, (1992), Dept. of Computer Science and Dept. of  \n  Mathematics and Stati
stics, James Cook University of North Queensland. \n  (Also submitted to Tec
hnometrics). \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 PE
RFORMANCE OF RDA" \n  Tech. Rep. no. 92-01, (1992), Dept. of Computer Scienc
e and Dept. of \n  Mathematics and Statistics, James Cook University of Nort
h 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']}
```

In [72]:

```
data.keys()
```

Out[72]:

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

In [73]:

```
data.feature_names
```

Out[73]:

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

In [74]:

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

Out[74]:

```
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]])
```

In [75]:

```
data.target
```

Out[75]:

```
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, 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])
```

In [76]:

```
columns_name = data.feature_names
```

In [77]:

```
columns_name
```

Out[77]:

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

In [78]:

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

In [79]:

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

In [80]:

```
df
```

Out[80]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

178 rows × 13 columns

In [81]:

```
data.target
```

Out[81]:

```
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, 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])
```

In [82]:

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

In [83]:

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

Out[83]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

178 rows × 14 columns

In [84]:

```
df.head()
```

Out[84]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavan
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [85]:

```
df.tail()
```

Out[85]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

In [86]:

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

Out[86]:

```
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
```

In [87]:

```
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
```

In [88]:

```
df.shape
```

Out[88]:

```
(178, 14)
```

In [89]:

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

In [90]:

```
X
```

Out[90]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	
...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	

178 rows × 13 columns

In [91]:

```
Y
```

Out[91]:

```
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
```

In [92]:

```
Y.head()
```

Out[92]:

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

In [93]:

```
# 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
```

Out[93]:

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

In [94]:

```
X_train
```

Out[94]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
22	13.71	1.86	2.36	16.6	101.0	2.61	2.88	
108	12.22	1.29	1.94	19.0	92.0	2.36	2.04	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	
145	13.16	3.57	2.15	21.0	102.0	1.50	0.55	
71	13.86	1.51	2.67	25.0	86.0	2.95	2.86	
...	...	...	...	...	...	...	...	
103	11.82	1.72	1.88	19.5	86.0	2.50	1.64	
67	12.37	1.17	1.92	19.6	78.0	2.11	2.00	
117	12.42	1.61	2.19	22.5	108.0	2.00	2.09	
47	13.90	1.68	2.12	16.0	101.0	3.10	3.39	
172	14.16	2.51	2.48	20.0	91.0	1.68	0.70	

124 rows × 13 columns

In [95]:

```
X_train.shape
```

Out[95]:

```
(124, 13)
```

In [96]:

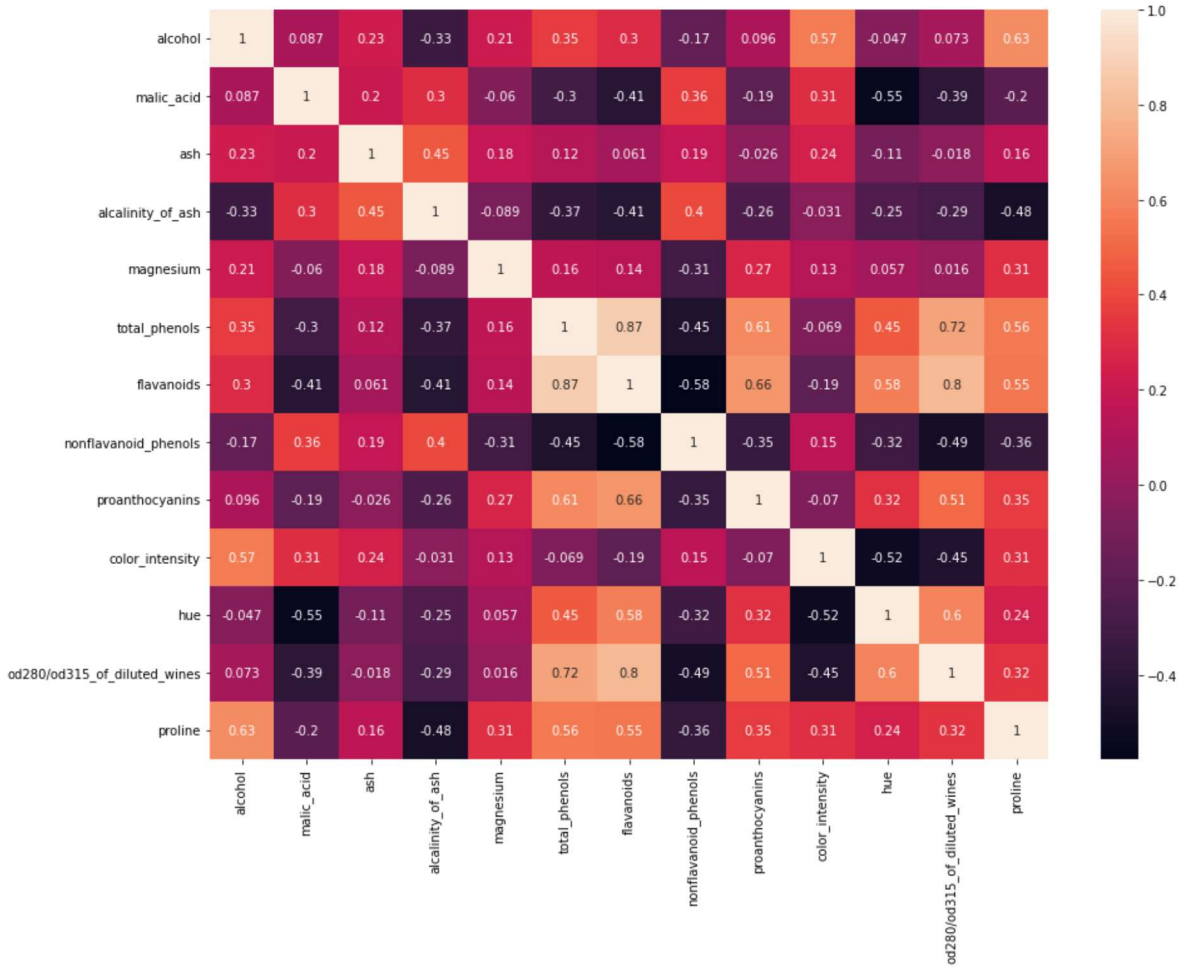
```
X_train.corr()
```

Out[96]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	t
alcohol	1.000000	0.087268	0.228809	-0.326030	0.212436	
malic_acid	0.087268	1.000000	0.200015	0.304109	-0.059823	
ash	0.228809	0.200015	1.000000	0.446093	0.181737	
alcalinity_of_ash	-0.326030	0.304109	0.446093	1.000000	-0.088590	
magnesium	0.212436	-0.059823	0.181737	-0.088590	1.000000	
total_phenols	0.352899	-0.298813	0.121369	-0.367199	0.163801	
flavanoids	0.296712	-0.408887	0.060808	-0.414673	0.143421	
nonflavanoid_phenols	-0.167773	0.363213	0.185052	0.398878	-0.305155	
proanthocyanins	0.095713	-0.190354	-0.025868	-0.255579	0.270090	
color_intensity	0.565029	0.305012	0.243573	-0.030653	0.125051	
hue	-0.047430	-0.545493	-0.108399	-0.251091	0.057459	
od280/od315_of_diluted_wines	0.073438	-0.390354	-0.018053	-0.287010	0.015833	
proline	0.627676	-0.200906	0.158194	-0.481131	0.312073	

In [97]:

```
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

In [98]:

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

In [99]:

```
arr[0][1]
```

Out[99]:

```
22
```

In [100]:

```
arr[0][0]
```

Out[100]:

```
11
```

In [101]:

```
arr[1][0]
```

Out[101]:

```
44
```

In [102]:

```
# Using for loop access the elements
```

In [103]:

```
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
```

In [104]:

```
# 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 co
                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
```

In [105]:

```
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', 'proanthocya
nins']
colname name which is correlated is color_intensity
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 'flavan
oids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 'hue',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity', 'hue']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 'hue',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity', 'hue', 'hue']
colname name which is correlated is hue
set {'color_intensity', 'proanthocyanins', 'nonflavanoid_phenols', 'hue',
'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', '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', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es']
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', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', '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', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', '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', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', 'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines', 'od28
0/od315_of_diluted_wines']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins', 'nonflavanoid_phenol
s', 'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', 'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines', 'od28
0/od315_of_diluted_wines', 'proline']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins', 'nonflavanoid_phenol
s', 'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', 'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines', 'od28
0/od315_of_diluted_wines', 'proline', 'proline']
colname name which is correlated is proline
set {'color_intensity', 'proline', 'proanthocyanins', 'nonflavanoid_phenol
s', 'hue', 'od280/od315_of_diluted_wines', 'flavanoids'}
lst ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanthocya
nins', 'color_intensity', 'hue', 'hue', 'hue', 'od280/od315_of_diluted_win
es', 'od280/od315_of_diluted_wines', 'od280/od315_of_diluted_wines', 'od28
0/od315_of_diluted_wines', 'proline', 'proline', 'proline']
list is ['flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'proanth
ocyanins', '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']
```

Out[105]:

```
7
```

In [106]:

```
corr_features
```

Out[106]:

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

In [107]:

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

In [108]:

```
X_train
```

Out[108]:

	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 × 6 columns

In [109]:

```
X_train.head()
```

Out[109]:

	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

In [110]:

```
X_train.tail()
```

Out[110]:

	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

In [111]:

```
X_test
```

Out[111]:

	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

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols
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

In [112]:

```
X_test.head()
```

Out[112]:

	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

In [113]:

```
X_test.tail()
```

Out[113]:

	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

In [114]:

```
df = df.drop(["flavanoids","nonflavanoid_phenols","proanthocyanins","color_intensity","hue"
df
```

Out[114]:

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

178 rows × 7 columns

Using the SVM Model

In [115]:

```
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
```

In [116]:

```
df
```

Out[116]:

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

178 rows × 7 columns

In [117]:

```
df.head()
```

Out[117]:

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

In [118]:

```
df.tail()
```

Out[118]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	target_values
173	13.71	5.65	2.45	20.5	95.0	1.68	2
174	13.40	3.91	2.48	23.0	102.0	1.80	2
175	13.27	4.28	2.26	20.0	120.0	1.59	2
176	13.17	2.59	2.37	20.0	120.0	1.65	2
177	14.13	4.10	2.74	24.5	96.0	2.05	2

In [119]:

```
feature_names = df[["alcohol","malic_acid","ash","alcalinity_of_ash","magnesium","total_phe
```

In [120]:

```
feature_names
```

Out[120]:

	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 × 6 columns

In [121]:

```
feature_names.dtypes
```

Out[121]:

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

In [122]:

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

Out[122]:

```
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 ]])
```

In [123]:

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

Out[123]:

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

Train/Test dataset

In [124]:

```
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)

In [125]:

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

Out[125]:

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

In [126]:

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

Out[126]:

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

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

In [127]:

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

Out[127]:

```
0.48148148148148157
```

In [128]:

```
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
```

In [129]:

```
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
```

In [130]:

```
df
```

Out[130]:

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

178 rows × 7 columns

In [131]:

```
X_test
```

Out[131]:

```
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 ]])
```

In [132]:

```
X_test.shape
```

Out[132]:

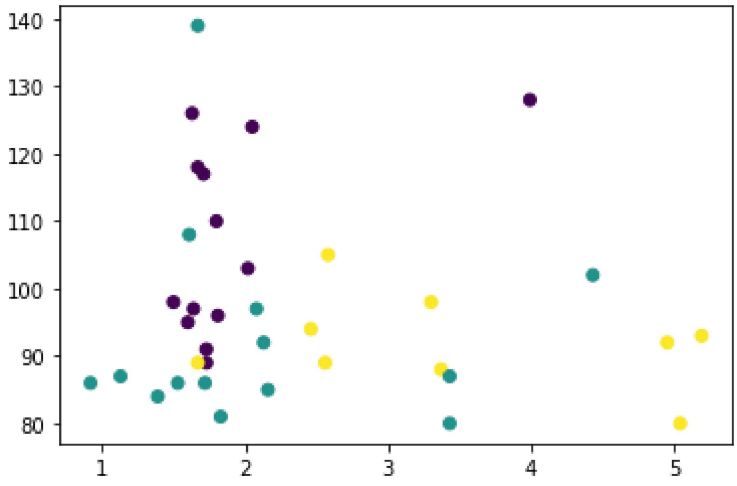
```
(36, 6)
```

In [133]:

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

Out[133]:

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

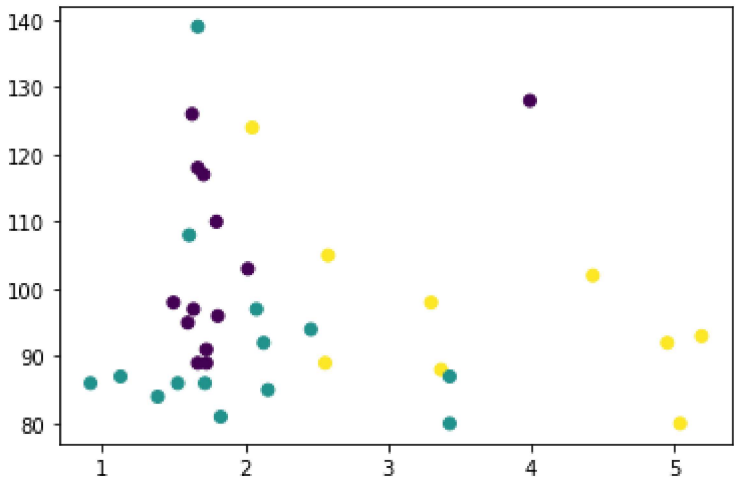


In [134]:

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

Out[134]:

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



In [135]:

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

Out[135]:

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

In [136]:

```
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
```

In [137]:

```
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

In [62]:

```
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
```

In [63]:

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

In [64]:

```
diabetes = load_diabetes()
```

In [65]:

```
diabetes
```

Out[65]:

```
{'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., 31
0., 101.,
         69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
         68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
         87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
        259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
        128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
        150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
        200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
         42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
         83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
        104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
        173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
        107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
         60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
        197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
         59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
        237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
        143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
        142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
         77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
         78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
        154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
         71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
        150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
        145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
         94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
         60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
         31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
        114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
        191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
        244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
        263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
         77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
         58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
        140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
        219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
         43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
        140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
         84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
         94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
        220.,  57.]),
```

```
 'frame': None,
 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTe
n baseline variables, age, sex, body mass index, average blood\npressure, an
d six blood serum measurements were obtained for each of n =\n442 diabetes p
atients, as well as the response of interest, a\nquantitative measure of dis
ease progression one year after baseline.\n\n**Data Set Characteristics:**\n
\n  :Number of Instances: 442\n\n  :Number of Attributes: First 10 columns a
re numeric predictive values\n\n  :Target: Column 11 is a quantitative measu
re of disease progression one year after baseline\n\n  :Attribute Informatio
n:\n      - age     age in years\n      - sex\n      - bmi     body mass ind
ex\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. t
he sum of squares of each column totals 1).\n\nSource URL:\nhttps://www4.sta
t.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBrad
ley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.\n(http
s://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.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'}
```


In [66]:

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

Out[66]:

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

In [67]:

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

Out[67]:

```
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]])
```

In [68]:

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

Out[68]:

```
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.])
```

In [69]:

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

Out[69]:

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

In [70]:

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

Out[70]:

```
'.. _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 progres
sion 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 p
redictive values\n\n  :Target: Column 11 is a quantitative measure of diseas
e 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-d
ensity lipoproteins\n      - s4      tch, total cholesterol / HDL\n      - s
5      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 su
m of squares of each column totals 1).\n\nSource URL:\nhttps://www4.stat.ncs
u.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley E
fron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angl
e Regression," Annals of Statistics (with discussion), 407-499.\n(https://we
b.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)'
```

In [71]:

```
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 ye
ar 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 (https://www4.sta
t.ncsu.edu/~boos/var.select/diabetes.html)

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

In [72]:

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

In [73]:

```
df
```

Out[73]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 10 columns

In [74]:

```
diabetes["target"]
```

Out[74]:

```
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.])
```

In [75]:

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

In [76]:

```
df
```

Out[76]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

442 rows × 11 columns

In [77]:

```
df.head()
```

Out[77]:

	age	sex	bmi	bp	s1	s2	s3	s4	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031

In [78]:

```
df.tail()
```

Out[78]:

	age	sex	bmi	bp	s1	s2	s3	s4	
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592	0.0
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309	-0.0
439	0.041708	0.050680	-0.015906	0.017282	-0.037344	-0.013840	-0.024993	-0.011080	-0.0
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560	0.0
441	-0.045472	-0.044642	-0.073030	-0.081414	0.083740	0.027809	0.173816	-0.039493	-0.0

In [79]:

```
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
```

In [80]:

```
df.describe()
```

Out[80]:

	age	sex	bmi	bp	s1	s2	4
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4
mean	-3.634285e-16	1.308343e-16	-8.045349e-16	1.281655e-16	-8.835316e-17	1.327024e-16	
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123996e-01	-1.267807e-01	-1.156131e-01	
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665645e-02	-3.424784e-02	-3.035840e-02	-3
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670611e-03	-4.320866e-03	-3.819065e-03	
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564384e-02	2.835801e-02	2.984439e-02	2
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320442e-01	1.539137e-01	1.987880e-01	1

In [81]:

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

Out[81]:

```
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
```

In [82]:

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

In [83]:

```
scaler = StandardScaler()
```

In [84]:

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

Out[84]:

```
StandardScaler()
```

In [85]:

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

In [86]:

```
scaled_data
```

Out[86]:

```
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]])
```

In [87]:

```
scaled_data.shape
```

Out[87]:

```
(442, 11)
```

In [88]:

```
### 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
```

In [89]:

```
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)
```

In [90]:

```
X_train.shape
```

Out[90]:

```
(309, 10)
```

In [91]:

```
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
```

Out[91]:

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

In [92]:

```
len(mutual_info)
```

Out[92]:

```
10
```

In [93]:

```
mutual_info
```

Out[93]:

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

In [94]:

```
X_train.columns
```

Out[94]:

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

In [95]:

```
mutual_info
```

Out[95]:

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

In [96]:

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

In [97]:

```
mutual_info
```

Out[97]:

```
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
```

In [98]:

```
type(mutual_info)
```

Out[98]:

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

In [99]:

```
mutual_info.index
```

Out[99]:

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

In [100]:

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

In [101]:

```
mutual_info
```

Out[101]:

```
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
```

In [102]:

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

Out[102]:

```
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
```

In [103]:

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

Out[103]:

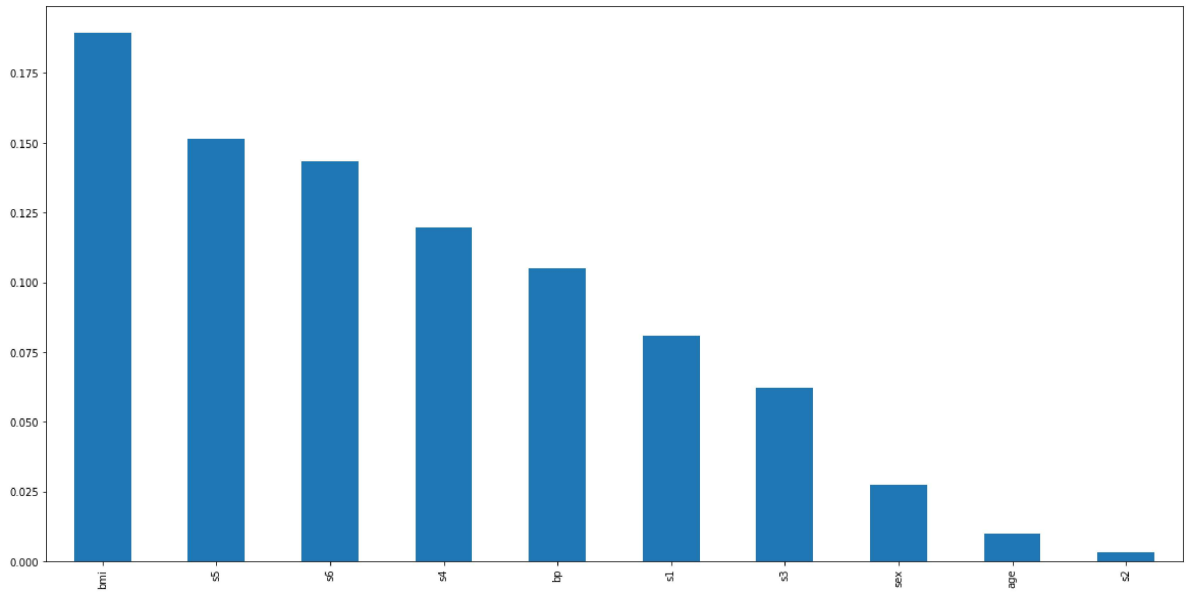
```
'age'
```

In [104]:

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

Out[104]:

```
<AxesSubplot:>
```



In [105]:

```
mutual_info[0]
```

Out[105]:

```
0.009802289238992401
```

In [106]:

```
mutual_info
```

Out[106]:

```
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
```

In [107]:

```
Y_train.shape
```

Out[107]:

```
(309,)
```

In [108]:

```
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
```

In [109]:

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

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

In [110]:

```
X_train
```

Out[110]:

	age	sex	bmi	bp	s1	s2	s3	s4	
232	0.012648	0.050680	0.000261	-0.011409	0.039710	0.057245	-0.039719	0.056081	0.0
224	-0.027310	-0.044642	-0.066563	-0.112400	-0.049727	-0.041397	0.000779	-0.039493	-0.0
252	0.005383	-0.044642	0.059541	-0.056166	0.024574	0.052861	-0.043401	0.050914	-0.0
254	0.030811	0.050680	0.056307	0.076958	0.049341	-0.012274	-0.036038	0.071210	0.1
418	0.009016	-0.044642	-0.024529	-0.026328	0.098876	0.094196	0.070730	-0.002592	-0.0
...	...	...	...	...	...	...	...	...	
323	0.070769	0.050680	-0.007284	0.049415	0.060349	-0.004445	-0.054446	0.108111	0.1
192	0.056239	0.050680	-0.030996	0.008101	0.019070	0.021233	0.033914	-0.039493	-0.0
117	0.059871	-0.044642	-0.021295	0.087287	0.045213	0.031567	-0.047082	0.071210	0.0
47	-0.078165	-0.044642	-0.073030	-0.057314	-0.084126	-0.074277	-0.024993	-0.039493	-0.0
172	0.041708	0.050680	0.071397	0.008101	0.038334	0.015909	-0.017629	0.034309	0.0

309 rows × 10 columns

In [111]:

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

In [112]:

```
X_train_columns
```

Out[112]:

	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 × 5 columns

Using the Decision Tree Model

In [180]:

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

In [181]:

```
X
```

Out[181]:

	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 × 5 columns

In [182]:

```
X.shape
```

Out[182]:

```
(309, 5)
```

In [185]:

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

In [186]:

```
Y
```

Out[186]:

```
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
```

In [187]:

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

Out[187]:

```
133
```

In [197]:

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

In [198]:

```
Y1
```

Out[198]:

```
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
```

In [199]:

```
Y1.shape
```

Out[199]:

```
(309,)
```

Setting up Decision Tree

In [200]:

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

In [201]:

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

In [202]:

```
X_train
```

Out[202]:

	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 × 5 columns

In [203]:

```
X_test
```

Out[203]:

	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 × 5 columns

In [204]:

```
Y1_train
```

Out[204]:

```
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
```

In [205]:

```
Y1_test
```

Out[205]:

```
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

In [206]:

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

Out[206]:

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

In [208]:

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

Out[208]:

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

In [220]:

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

In [221]:

```
y_pred
```

Out[221]:

```
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.])
```

In [222]:

```
X_test.shape
```

Out[222]:

```
(93, 5)
```

In [223]:

```
Y1_test.shape
```

Out[223]:

```
(93,)
```

In [224]:

```
y_pred.shape
```

Out[224]:

```
(93,)
```

In [225]:

```
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
```

In [226]:

```
Y1_test
```

Out[226]:

```
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
```

In [227]:

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

In [228]:

```
Y1_test.shape
```

Out[228]:

```
(93,)
```

In [229]:

```
y_pred.shape
```

Out[229]:

```
(93,)
```

In [230]:

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

```
R2 Score : 100.0 %
```

In [232]:

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

In [233]:

```
accuracy
```

Out[233]:

```
'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

In [179]:

```
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
```

In [180]:

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

In [181]:

```
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 deprecat
ed 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 th
is
    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)
```

In [182]:

```
boston_df
```

Out[182]:

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690
e+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 Instance
s: 506 \n\n    :Number of Attributes: 13 numeric/categorical predictive. Med
ian Value (attribute 14) is usually the target.\n\n    :Attribute Informatio
n (in order):\n        - CRIM     per capita crime rate by town\n        - Z
N       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 t
o 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 o
f 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 housi
ng dataset.\nhttps://archive.ics.uci.edu/ml/machine-learning-databases/housi
ng/\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. En
viron. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley, Kuh
& Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transfo
rmations are used in the table on\npages 244-261 of the latter.\n\nThe Bosto
n house-price data has been used in many machine learning papers that addres
s regression\nproblems.   \n     \n.. topic:: References\n\n   - Belsley, Ku
h & Welsch, 'Regression diagnostics: Identifying Influential Data and Source
s of Collinearity', Wiley, 1980. 244-261.\n   - Quinlan,R. (1993). Combining
Instance-Based and Model-Based Learning. In Proceedings on the Tenth Interna
tional Conference of Machine Learning, 236-243, University of Massachusetts,
Amherst. Morgan Kaufmann.\n",
 'filename': 'boston_house_prices.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [183]:

```
boston_df.data
```

Out[183]:

```
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]])
```

In [184]:

```
boston_df.keys()
```

Out[184]:

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

In [185]:

```
boston_df.feature_names
```

Out[185]:

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

In [186]:

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

Out[186]:

```
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])
```

In [187]:

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

In [188]:

```
df
```

Out[188]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 13 columns

In [189]:

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

In [190]:

```
df
```

Out[190]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90
...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90

506 rows × 14 columns

In [191]:

```
df.head()
```

Out[191]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

In [192]:

```
df.tail()
```

Out[192]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	

In [193]:

```
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
```

In [194]:

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

Out[194]:

```
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
```

In [195]:

```
df.describe()
```

Out[195]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	50
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	1

In [196]:

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

In [197]:

```
scaler = StandardScaler()
```

In [198]:

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

Out[198]:

```
StandardScaler()
```

In [199]:

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

In [200]:

```
scaled_data
```

Out[200]:

```
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]])
```

In [201]:

```
scaled_data.shape
```

Out[201]:

```
(506, 14)
```

Train and Test Split

In [202]:

```
### 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
```

In [203]:

```
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)
```

In [204]:

```
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
```

Out[204]:

```
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 ])
```

In [205]:

```
mutual_info
```

Out[205]:

```
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 ])
```

In [206]:

```
X_train.columns
```

Out[206]:

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

In [207]:

```
mutual_info
```

Out[207]:

```
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 ])
```

In [208]:

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

In [209]:

```
mutual_info
```

Out[209]:

```
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
```

In [210]:

```
X_train.columns
```

Out[210]:

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

In [211]:

```
type(mutual_info)
```

Out[211]:

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

In [212]:

```
mutual_info.index
```

Out[212]:

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

In [213]:

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

In [214]:

```
mutual_info
```

Out[214]:

```
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
```

In [215]:

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

Out[215]:

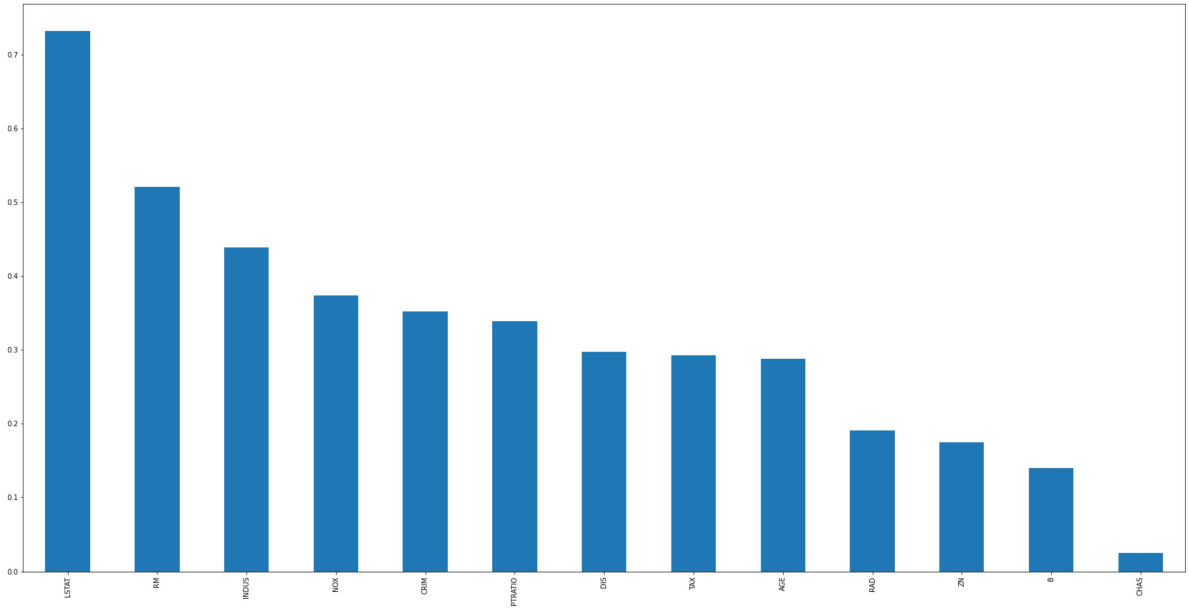
```
LSTAT      0.731849
RM         0.521005
INDUS      0.438355
NOX        0.373918
CRIM       0.351748
PTRATIO    0.338427
DIS        0.296951
TAX        0.292781
AGE        0.287936
RAD        0.190633
ZN         0.175092
B          0.139642
CHAS       0.025265
dtype: float64
```

In [216]:

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

Out[216]:

<AxesSubplot:>



In [217]:

```
mutual_info[0]
```

Out[217]:

0.35174779022450053

In [218]:

```
mutual_info
```

Out[218]:

```
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
```

In [219]:

```
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
```

In [220]:

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

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

In [221]:

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

In [222]:

```
X_train_columns
```

Out[222]:

	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 × 6 columns

In [223]:

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

In [224]:

```
X
```

Out[224]:

```
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]])
```

In [225]:

```
X.shape
```

Out[225]:

```
(506, 6)
```

In [233]:

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

In [234]:

```
Y.shape
```

Out[234]:

```
(506, 1)
```

In [235]:

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

Out[235]:

```
target_values    0
dtype: int64
```

Splitting the dataset into the Training set and Test set

In [236]:

```
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 = 1
```

In [237]:

```
X_train.shape
```

Out[237]:

```
(404, 6)
```

In [238]:

```
X_test.shape
```

Out[238]:

```
(102, 6)
```

Feature Scaling

In [239]:

```
# 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)
```

In [240]:

```
X_train
```

Out[240]:

```
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]])
```

In [241]:

```
Y_train
```

Out[241]:

```
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],
```

In [242]:

```
X_test
```

Out[242]:

```
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],
```

In [243]:

```
Y_test
```

Out[243]:

```
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

In [244]:

```
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]]
```

In [245]:

```
regr.intercept_
```

Out[245]:

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

In [246]:

```
regr.coef_
```

Out[246]:

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

In [247]:

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

Out[247]:

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

In [250]:

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

Out[250]:

```
0.021912354709027443
```

In [251]:

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

Out[251]:

```
-0.041174713622634834
```

In [252]:

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

In [253]:

```
y_pred
```

Out[253]:

```
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]])
```

In [254]:

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

In [255]:

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

```
R2 Score : 70.70768805651036 %
```

In [256]:

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

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

In [257]:

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

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

In [258]:

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

In [259]:

```
accuracy
```

Out[259]:

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

Apply Lasso Regression to cover the overfitting and underfitting problem

In [260]:

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

In [261]:

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

In [262]:

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

Out[262]:

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

In [263]:

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

In [264]:

```
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% ¶