ilearn-healthcare-capstone-project

August 14, 2023

0.0.1 Importing Libraries

[203]:
```
%matplotlib inline
import numpy as np
import pandas as pd


import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
```

0.0.2 Loading Dataset

[204]:
```
data = pd.read_csv('health care diabetes.csv')
```

[205]:
```
data.head()
```

[205]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[206]:
```
data.shape
```

[206]:
```
(768, 9)
```

0.1 Project Task: Week 1 – Data Exploration and Missing Values Treatment

[304]:
```
#Checking for null values in Dataset
data.isnull().any()
```

[304]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

Since the 0 value in Glucose,BloodPressure,SkinThickness,Insulin and BMI variables represent missing values.Lets find now many instances are there in each of the above variables

[208]:
```
data[data['Glucose']==0]
```

[208]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
75             1        0             48             20        0  24.7
182            1        0             74             20       23  27.7
342            1        0             68             35        0  32.0
349            5        0             80             32        0  41.0
502            6        0             68             41        0  39.0

     DiabetesPedigreeFunction  Age  Outcome
75                      0.140   22        0
182                     0.299   21        0
342                     0.389   22        0
349                     0.346   37        1
502                     0.727   41        1
```

[209]:
```
(5/765)*100
#only 0.6% of data is having missing values in Glucose column. No need to worry␣
↪we can ignore them
```

[209]:
```
0.6535947712418301
```

[210]:
```
(data[data['BloodPressure']==0]).shape
```

[210]:
```
(35, 9)
```

[211]:
```
(35/765)*100
#4.5% of data is having missing values in BloodPressure column
```

[211]: 4.57516339869281

[212]:
```
(data[data['SkinThickness']==0]).shape
```

[212]: (227, 9)

[213]:
```
(227/765)*100
#29.6% of data is having missing values in SkinThickness column
```

[213]: 29.673202614379086

[214]:
```
(data[data['Insulin']==0]).shape
```

[214]: (374, 9)

[215]:
```
(374/765)*100
#~49% of data is having missing values in Insulin column
```

[215]: 48.888888888888886

[216]:
```
(data[data['BMI']==0]).shape
```

[216]: (11, 9)

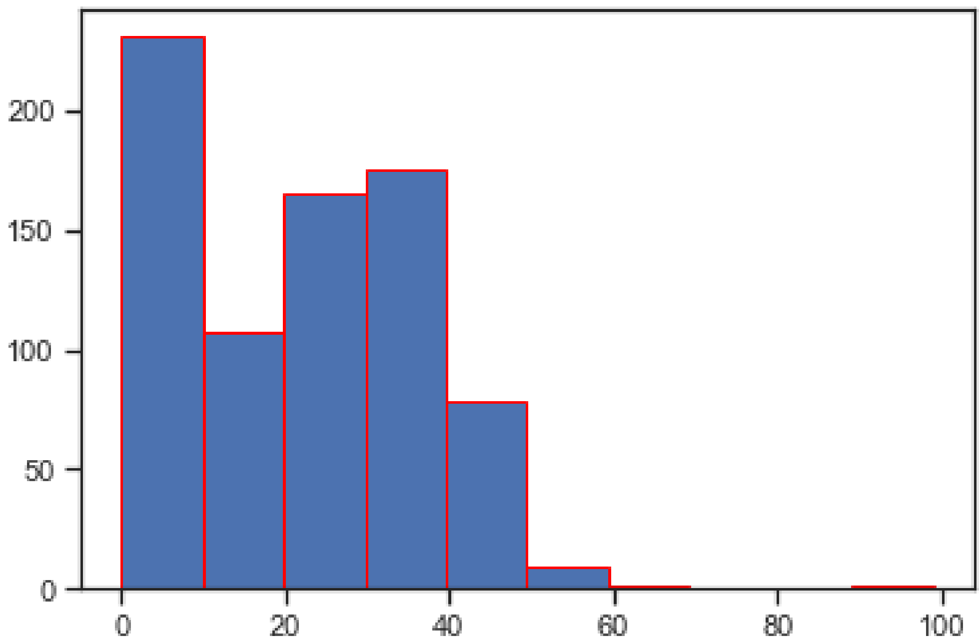
[217]:
```
(11/765)*100
#1.4% of data is having missing values in BMI column
```

[217]: 1.4379084967320261

Since Insulin and SkinThickness are having higher percentages of missing values lets try to fill up the missing values

[218]:
```
plt.hist(data['SkinThickness'],edgecolor='red')
```

[218]:
```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```

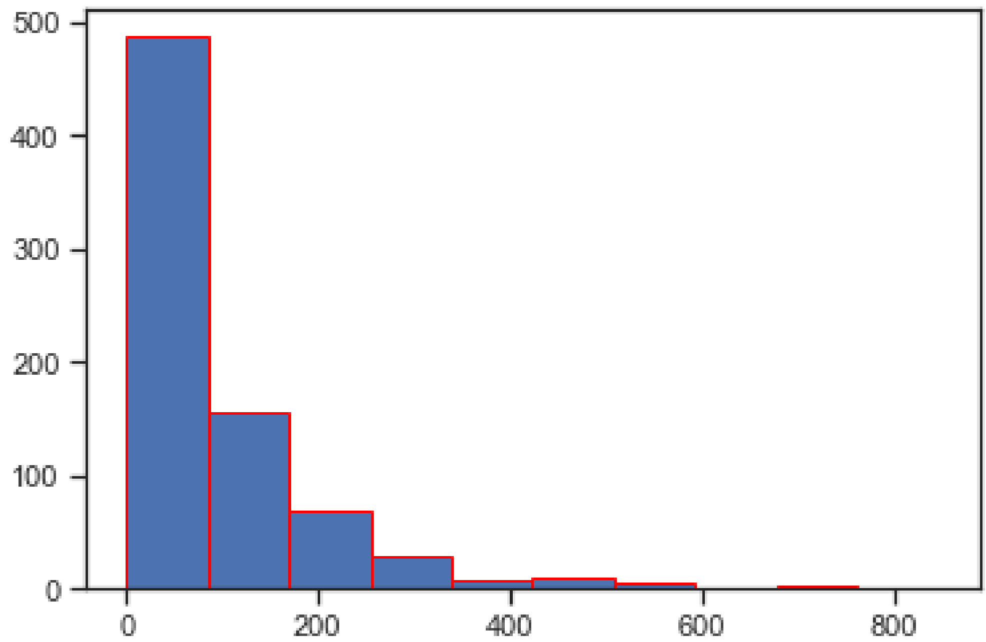


[219]:
```
data[data['SkinThickness']!=0]['SkinThickness'].describe()
```

[219]:
```
count    541.000000
mean      29.153420
std       10.476982
min        7.000000
25%       22.000000
50%       29.000000
75%       36.000000
max       99.000000
Name: SkinThickness, dtype: float64
```

[220]:
```
plt.hist(data['Insulin'],edgecolor='red')
```

[220]:
```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <BarContainer object of 10 artists>)
```



[221]:
```
data[data['Insulin']!=0]['Insulin'].describe()
```

[221]:
```
count    394.000000
mean     155.548223
std      118.775855
min       14.000000
25%       76.250000
50%      125.000000
75%      190.000000
max      846.000000
Name: Insulin, dtype: float64
```

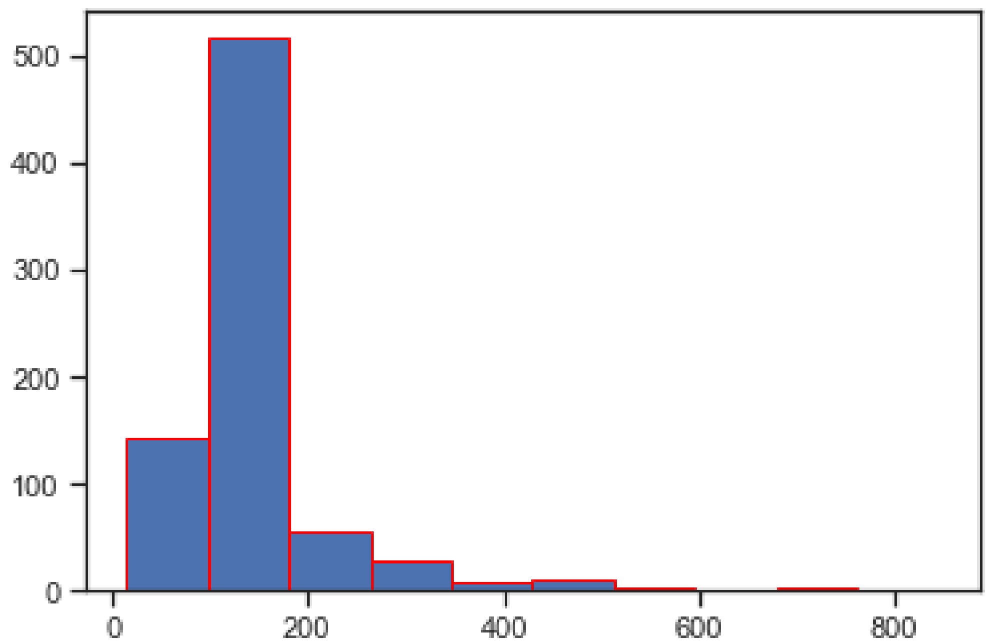
Mean value of Skinthickness is ~29 and the mean value of Insulin is ~155 let impute the missing values with means

[222]:
```
from numpy import nan
dataset_imputed = data
dataset_imputed[['SkinThickness','Insulin']] =␣
 ↪dataset_imputed[['SkinThickness','Insulin']].replace(0, nan)
```

[223]:
```
dataset_imputed.fillna(dataset_imputed.mean(), inplace=True)
```

[224]:
```
plt.hist(dataset_imputed['Insulin'],edgecolor='red')
```

[224]: (array([142., 517., 55., 29.,   7., 10.,   4.,   1.,   2.,   1.]),
 array([14. , 97.2, 180.4, 263.6, 346.8, 430. , 513.2, 596.4, 679.6,
        762.8, 846.]),
 <BarContainer object of 10 artists>)



[225]:
```
data.describe()
```

[225]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	29.153420	155.548223
std	3.369578	31.972618	19.355807	8.790942	85.021108
min	0.000000	0.000000	0.000000	7.000000	14.000000
25%	1.000000	99.000000	62.000000	25.000000	121.500000
50%	3.000000	117.000000	72.000000	29.153420	155.548223
75%	6.000000	140.250000	80.000000	32.000000	155.548223
max	17.000000	199.000000	122.000000	99.000000	846.000000

\

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000

```
max     67.100000                   2.420000   81.000000    1.000000
```

[226]:

```
dataset_imputed.describe()
```

[226]:

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      29.153420  155.548223
std       3.369578   31.972618      19.355807       8.790942   85.021108
min       0.000000    0.000000       0.000000       7.000000   14.000000
25%       1.000000   99.000000      62.000000      25.000000  121.500000
50%       3.000000  117.000000      72.000000      29.153420  155.548223
75%       6.000000  140.250000      80.000000      32.000000  155.548223
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[227]:

```
dataset_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(4), int64(5)
memory usage: 54.1 KB
```

[228]:

```
Positive = dataset_imputed[dataset_imputed['Outcome']==1]
Positive.head(5)
```

[228]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0            6      148             72       35.00000  155.548223  33.6
2            8      183             64       29.15342  155.548223  23.3
4            0      137             40       35.00000  168.000000  43.1
6            3       78             50       32.00000   88.000000  31.0
8            2      197             70       45.00000  543.000000  30.5

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
2                     0.672   32        1
4                     2.288   33        1
6                     0.248   26        1
8                     0.158   53        1
```

[229]:
```
Negative = dataset_imputed[dataset_imputed['Outcome']==0]
Negative.head(5)
```

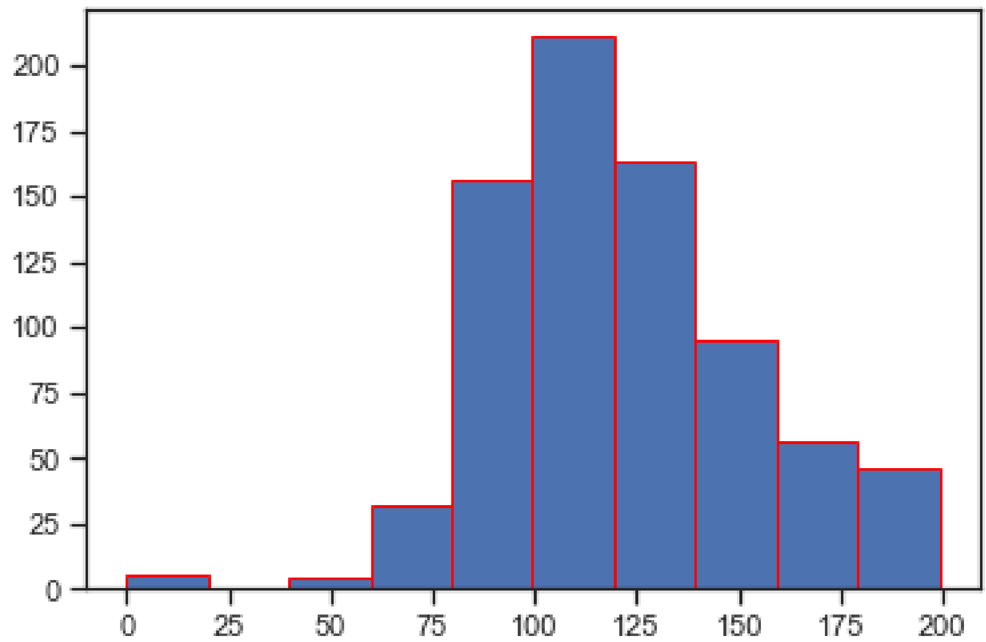
[229]:
```
    Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
1             1       85             66       29.00000  155.548223  26.6
3             1       89             66       23.00000   94.000000  28.1
5             5      116             74       29.15342  155.548223  25.6
7            10      115              0       29.15342  155.548223  35.3
10            4      110             92       29.15342  155.548223  37.6

    DiabetesPedigreeFunction  Age  Outcome
1                      0.351   31        0
3                      0.167   21        0
5                      0.201   30        0
7                      0.134   29        0
10                     0.191   30        0
```

[230]:
```
dataset_imputed['Glucose'].value_counts().head(5)
```

[230]:
```
99     17
100    17
111    14
129    14
125    14
Name: Glucose, dtype: int64
```

[231]:
```
plt.hist(dataset_imputed['Glucose'],edgecolor='red')
```

[231]:
```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```

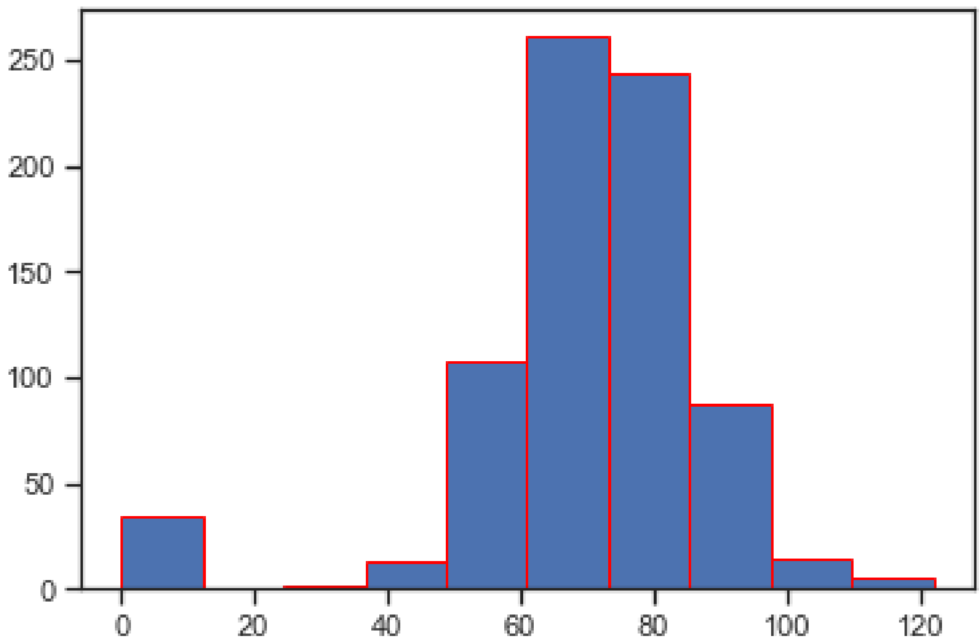


[232]:
```
dataset_imputed['BloodPressure'].value_counts().head(7)
```

[232]:
```
70    57
74    52
78    45
68    45
72    44
64    43
80    40
Name: BloodPressure, dtype: int64
```

[233]:
```
plt.hist(dataset_imputed['BloodPressure'],edgecolor='red')
```

[233]:
```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <BarContainer object of 10 artists>)
```

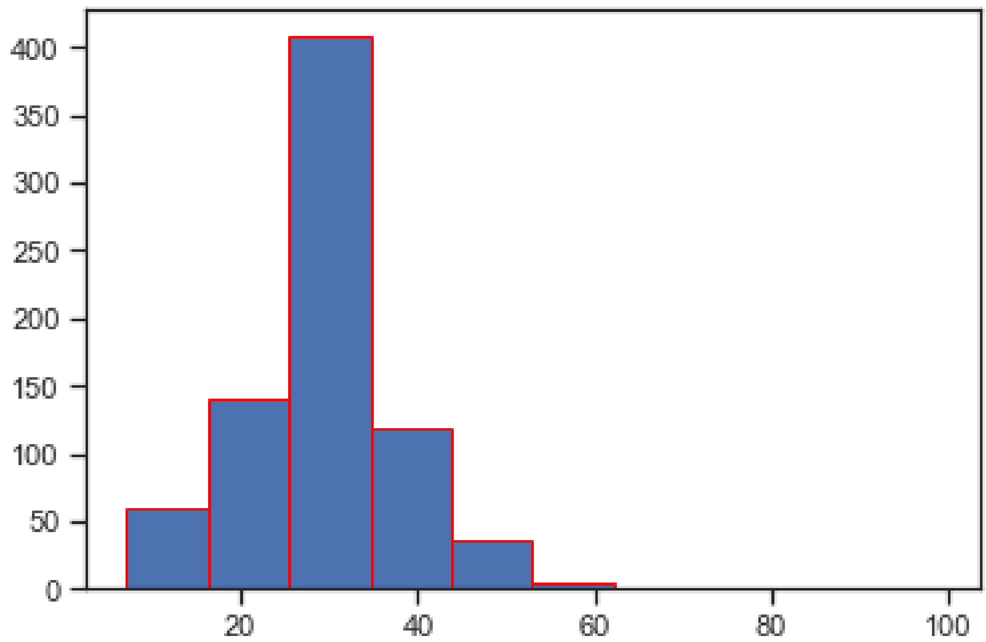


[234]:
```
dataset_imputed['SkinThickness'].value_counts().head(7)
```

[234]:
```
29.15342    227
32.00000     31
30.00000     27
27.00000     23
23.00000     22
33.00000     20
28.00000     20
Name: SkinThickness, dtype: int64
```

[235]:
```
plt.hist(dataset_imputed['SkinThickness'],edgecolor='red')
```

[235]:
```
(array([ 59., 141., 408., 118.,  36.,   4.,   1.,   0.,   0.,   1.]),
 array([ 7. , 16.2, 25.4, 34.6, 43.8, 53. , 62.2, 71.4, 80.6, 89.8, 99. ]),
 <BarContainer object of 10 artists>)
```

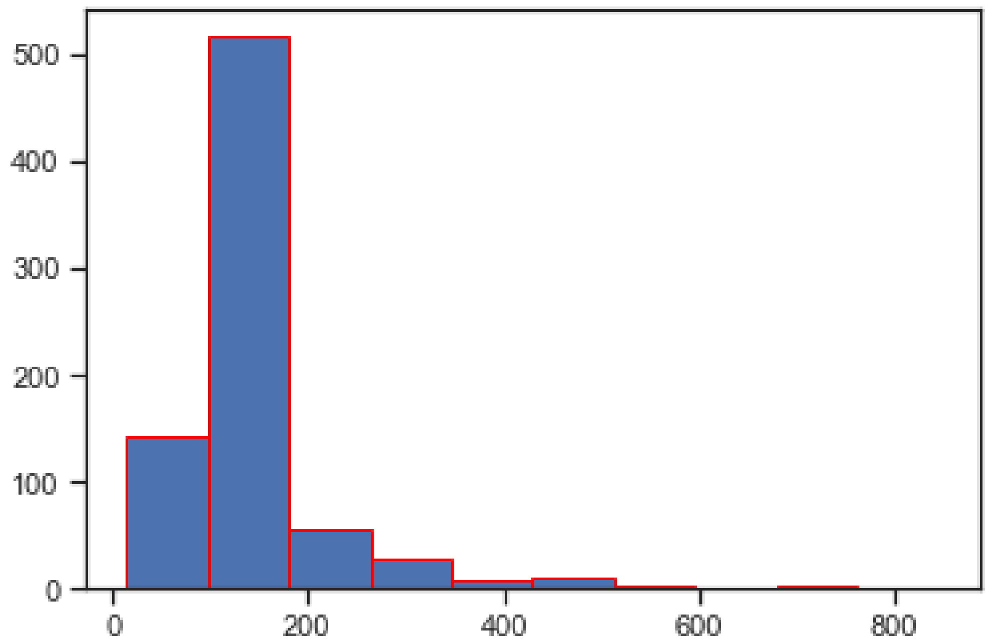


[236]:
```
dataset_imputed['Insulin'].value_counts().head(7)
```

[236]:
```
155.548223    374
105.000000     11
130.000000      9
140.000000      9
120.000000      8
94.000000       7
180.000000      7
Name: Insulin, dtype: int64
```

[237]:
```
plt.hist(dataset_imputed['Insulin'],edgecolor='red')
```

[237]:
```
(array([142., 517.,  55.,  29.,   7.,  10.,   4.,   1.,   2.,   1.]),
 array([ 14. ,  97.2, 180.4, 263.6, 346.8, 430. , 513.2, 596.4, 679.6,
        762.8, 846. ]),
 <BarContainer object of 10 artists>)
```

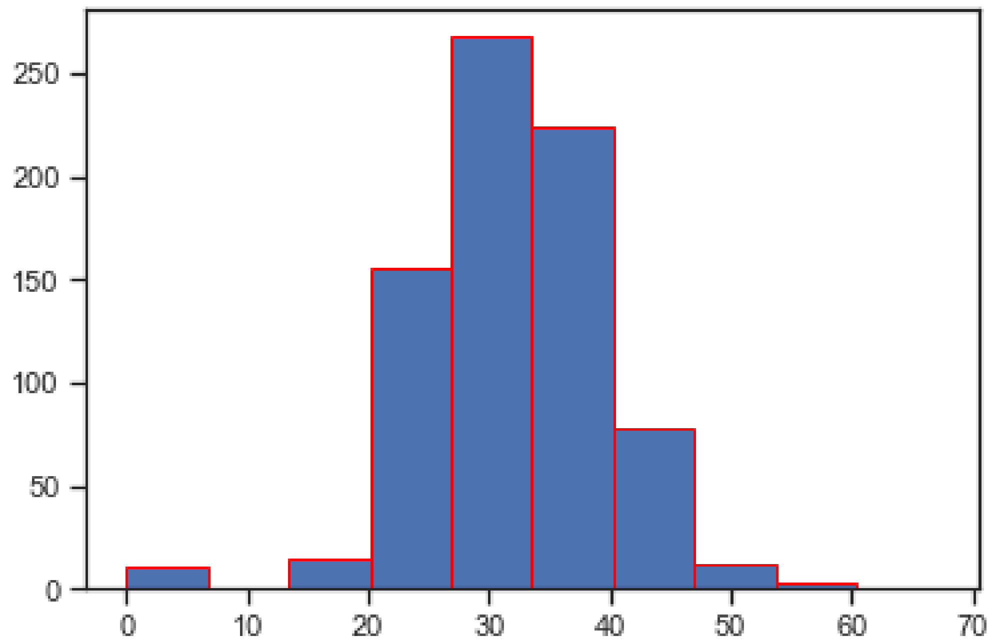


[238]:
```
dataset_imputed['BMI'].value_counts().head(7)
```

[238]:
```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
33.3    10
30.1     9
Name: BMI, dtype: int64
```

[239]:
```
plt.hist(dataset_imputed['BMI'],edgecolor='red')
```

[239]:
```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <BarContainer object of 10 artists>)
```



[240]:
```
dataset_imputed.describe().transpose()
```

[240]:
```
                          count        mean        std     min        25%  \
Pregnancies               768.0    3.845052   3.369578   0.000    1.00000
Glucose                   768.0  120.894531  31.972618   0.000   99.00000
BloodPressure             768.0   69.105469  19.355807   0.000   62.00000
SkinThickness             768.0   29.153420   8.790942   7.000   25.00000
Insulin                   768.0  155.548223  85.021108  14.000  121.50000
BMI                       768.0   31.992578   7.884160   0.000   27.30000
DiabetesPedigreeFunction  768.0    0.471876   0.331329   0.078    0.24375
Age                       768.0   33.240885  11.760232  21.000   24.00000
Outcome                   768.0    0.348958   0.476951   0.000    0.00000

                                 50%         75%     max
Pregnancies                 3.000000    6.000000   17.00
Glucose                   117.000000  140.250000  199.00
BloodPressure              72.000000   80.000000  122.00
SkinThickness              29.153420   32.000000   99.00
Insulin                   155.548223  155.548223  846.00
BMI                        32.000000   36.600000   67.10
DiabetesPedigreeFunction    0.372500    0.626250    2.42
Age                        29.000000   41.000000   81.00
Outcome                     0.000000    1.000000    1.00
```

0.1.1 Project Task: Week 2 – Corelation Analysis and Scatter Plots

[241]:
```
Positive.shape
```

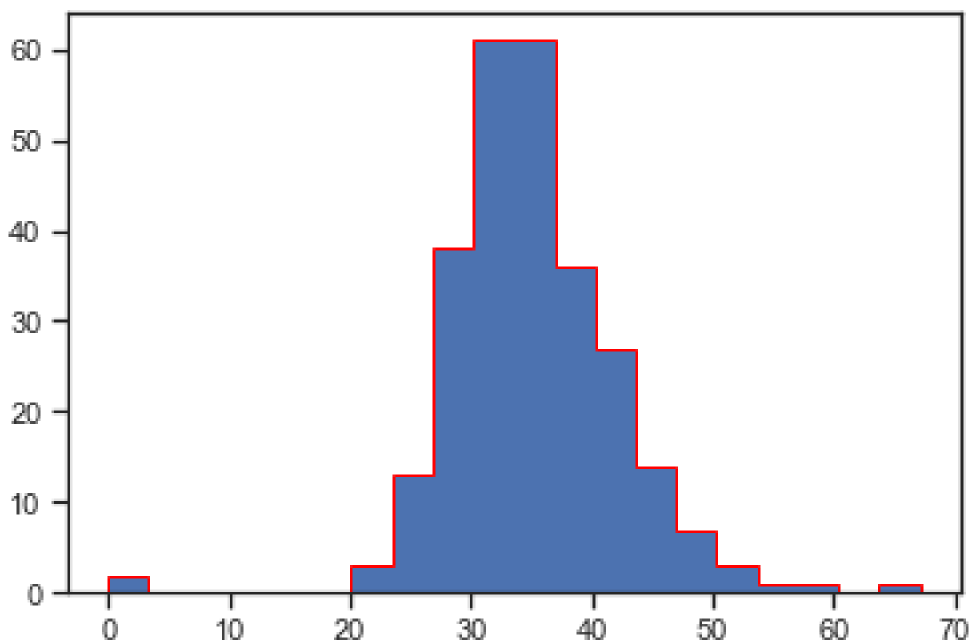
[241]:
```
(268, 9)
```

[242]:
```
Negative.shape
```

[242]:
```
(500, 9)
```

[243]:
```
plt.hist(Positive['BMI'],histtype='stepfilled',bins=20,edgecolor='red')
```

[243]:
```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  3., 13., 38., 61., 61., 36., 27.,
        14.,  7.,  3.,  1.,  1.,  0.,  1.]),
 array([ 0.   ,  3.355,  6.71 , 10.065, 13.42 , 16.775, 20.13 , 23.485,
        26.84 , 30.195, 33.55 , 36.905, 40.26 , 43.615, 46.97 , 50.325,
        53.68 , 57.035, 60.39 , 63.745, 67.1  ]),
 [<matplotlib.patches.Polygon at 0x7f93bbaaa640>])
```



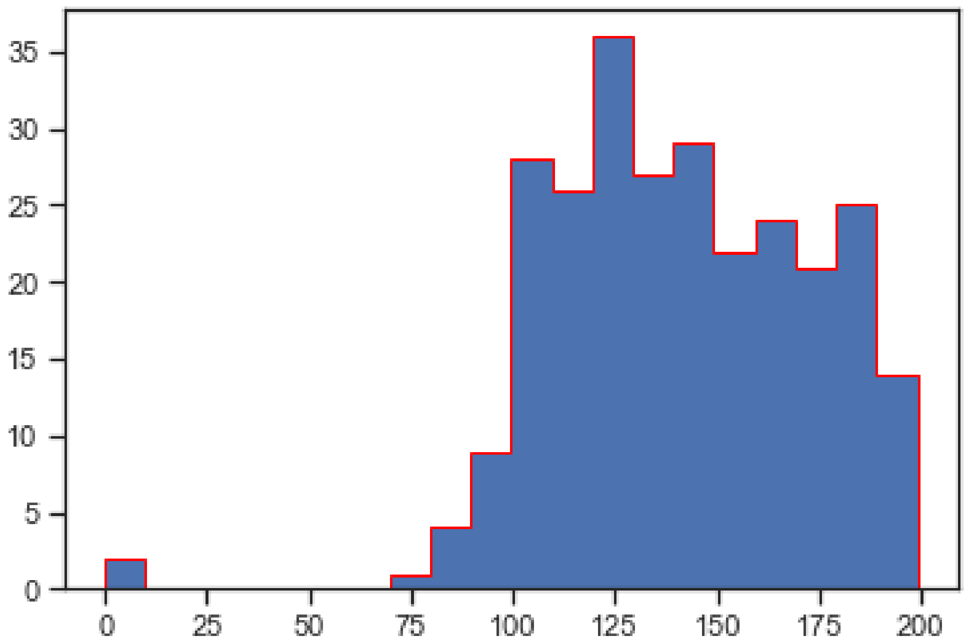
[244]:
```
Positive['BMI'].value_counts().head(7)
```

[244]:
```
32.9    8
31.6    7
33.3    6
31.2    5
30.5    5
```

```
32.0    5
34.3    4
Name: BMI, dtype: int64
```

[245]:
```
plt.hist(Positive['Glucose'],histtype='stepfilled',bins=20,edgecolor='red')
```

[245]:
```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  4.,  9., 28., 26., 36.,
        27., 29., 22., 24., 21., 25., 14.]),
 array([  0.  ,   9.95,  19.9 ,  29.85,  39.8 ,  49.75,  59.7 ,  69.65,
         79.6 ,  89.55,  99.5 , 109.45, 119.4 , 129.35, 139.3 , 149.25,
        159.2 , 169.15, 179.1 , 189.05, 199.  ]),
 [<matplotlib.patches.Polygon at 0x7f93bb3d4490>])
```

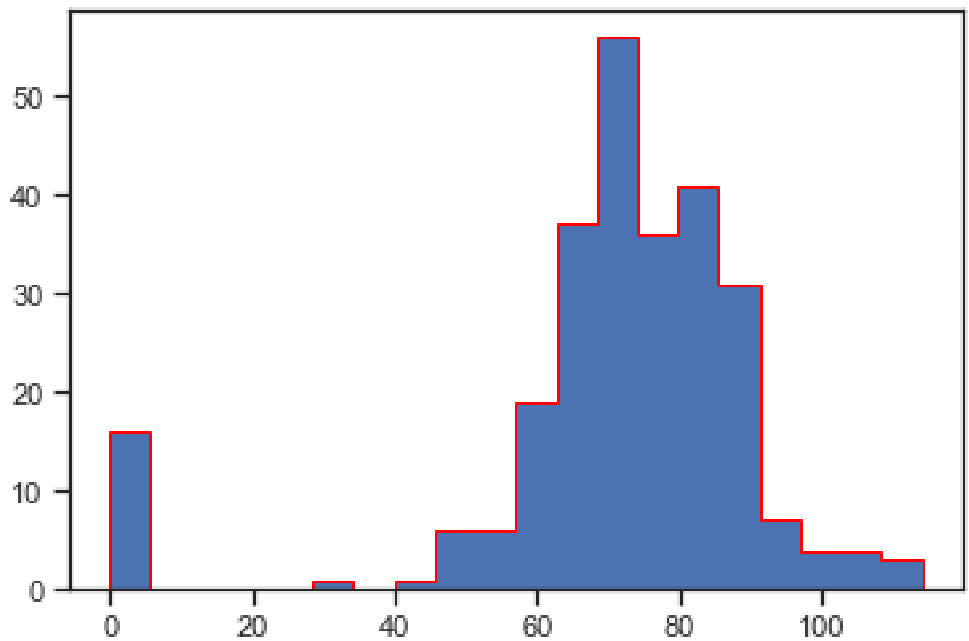


[246]:
```
Positive['Glucose'].value_counts().head(7)
```

[246]:
```
125    7
128    6
129    6
115    6
158    6
146    5
124    5
Name: Glucose, dtype: int64
```

[247]:
```
plt.
↪hist(Positive['BloodPressure'],histtype='stepfilled',bins=20,edgecolor='red')
```

[247]:
```
(array([16.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,  6.,  6., 19., 37., 56.,
        36., 41., 31.,  7.,  4.,  4.,  3.]),
 array([  0. ,   5.7,  11.4,  17.1,  22.8,  28.5,  34.2,  39.9,  45.6,
         51.3,  57. ,  62.7,  68.4,  74.1,  79.8,  85.5,  91.2,  96.9,
        102.6, 108.3, 114. ]),
 [<matplotlib.patches.Polygon at 0x7f93d4849e50>])
```

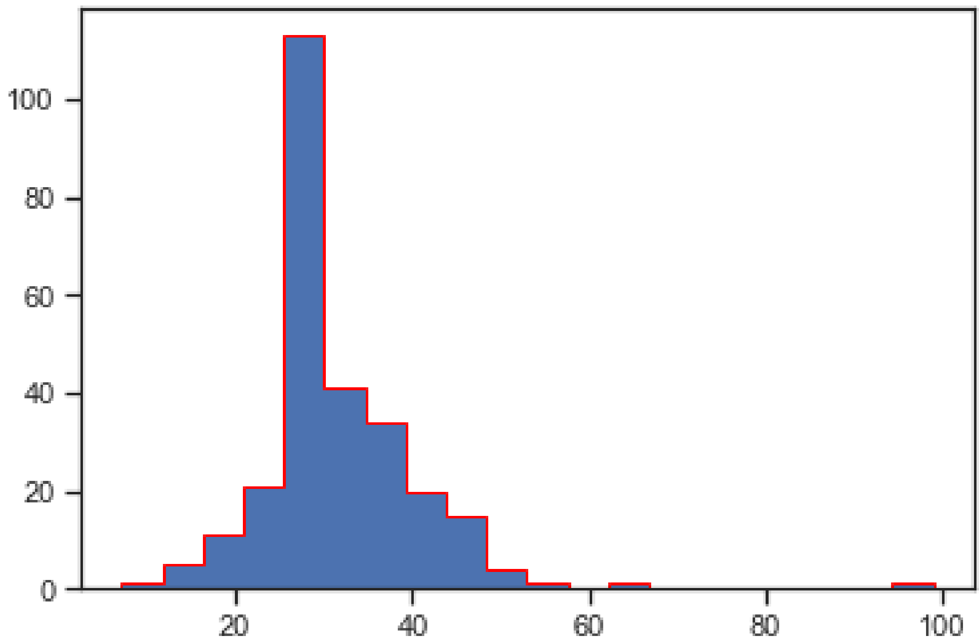


[248]:
```
Positive['BloodPressure'].value_counts().head(7)
```

[248]:
```
70    23
76    18
78    17
74    17
72    16
0     16
80    13
Name: BloodPressure, dtype: int64
```

[249]:
```
plt.
↪hist(Positive['SkinThickness'],histtype='stepfilled',bins=20,edgecolor='red')
```

```
[249]: (array([  1.,   5.,  11.,  21., 113.,  41.,  34.,  20.,  15.,   4.,   1.,
          0.,   1.,   0.,   0.,   0.,   0.,   0.,   0.,   1.]),
 array([ 7. , 11.6, 16.2, 20.8, 25.4, 30. , 34.6, 39.2, 43.8, 48.4, 53. ,
        57.6, 62.2, 66.8, 71.4, 76. , 80.6, 85.2, 89.8, 94.4, 99. ]),
 [<matplotlib.patches.Polygon at 0x7f93d2eb1940>])
```

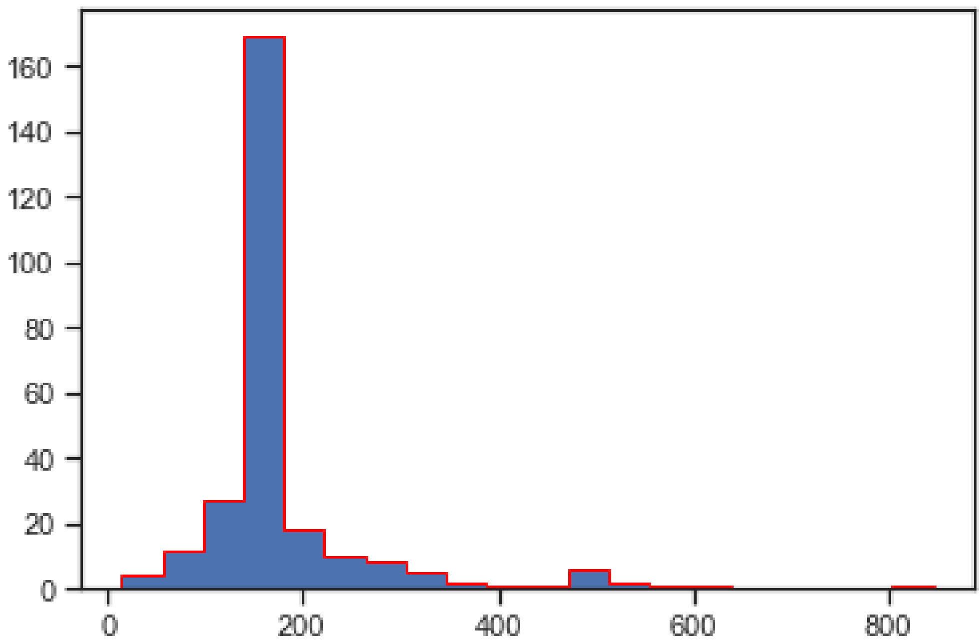



```
[250]: Positive['SkinThickness'].value_counts().head(7)
```

```
[250]: 29.15342    88
32.00000    14
30.00000     9
33.00000     9
39.00000     8
37.00000     8
36.00000     8
Name: SkinThickness, dtype: int64
```

```
[251]: plt.hist(Positive['Insulin'],histtype='stepfilled',bins=20,edgecolor='red')
```

```
[251]: (array([  4.,  12.,  27., 169.,  18.,  10.,   8.,   5.,   2.,   1.,   1.,
          6.,   2.,   1.,   1.,   0.,   0.,   0.,   0.,   1.]),
 array([ 14. ,  55.6,  97.2, 138.8, 180.4, 222. , 263.6, 305.2, 346.8,
        388.4, 430. , 471.6, 513.2, 554.8, 596.4, 638. , 679.6, 721.2,
        762.8, 804.4, 846. ]),
 [<matplotlib.patches.Polygon at 0x7f93d449bf40>])
```



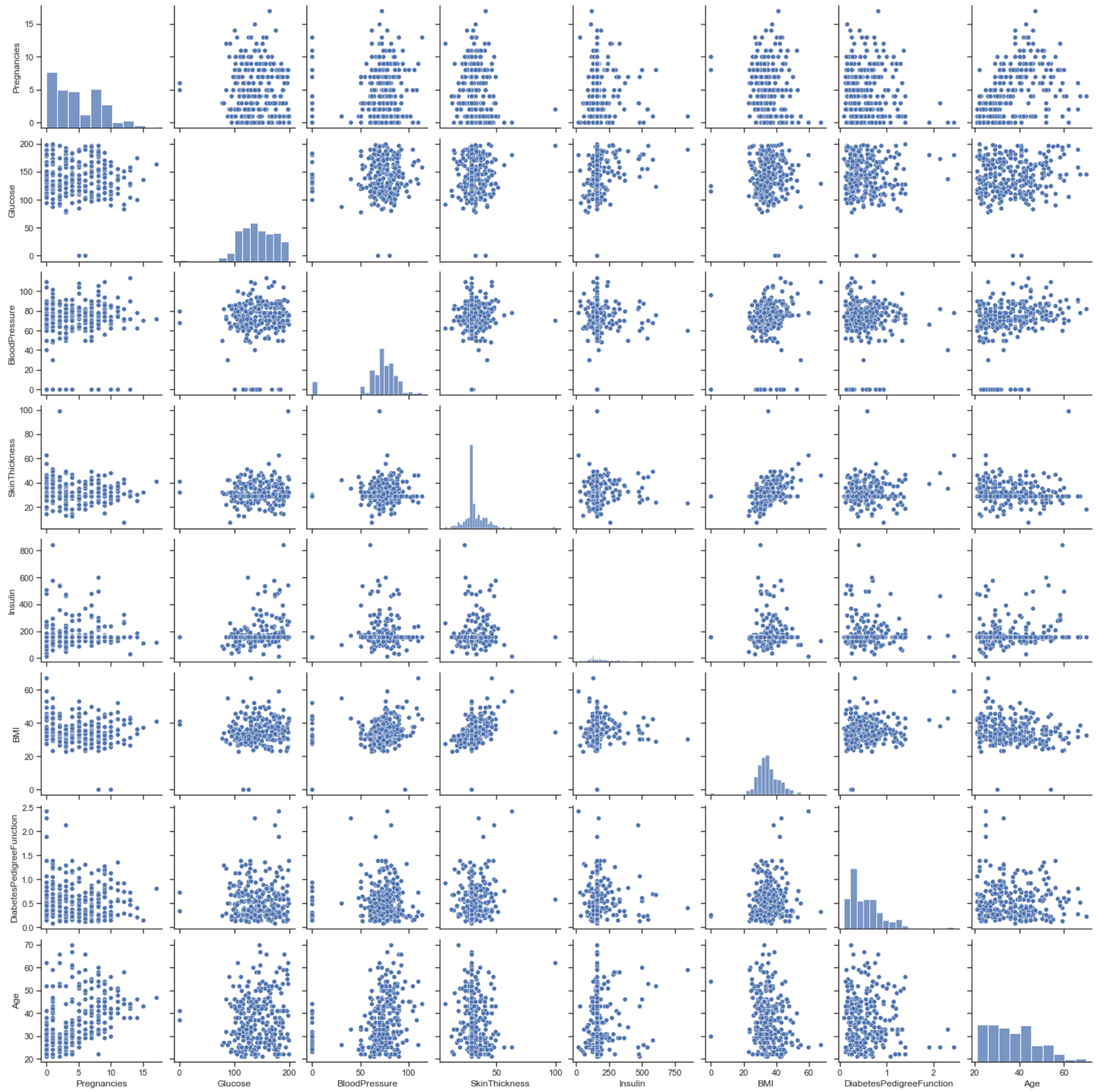
[252]:
```
Positive['Insulin'].value_counts().head(7)
```

[252]:
```
155.548223    138
130.000000      6
180.000000      4
175.000000      3
156.000000      3
185.000000      2
194.000000      2
Name: Insulin, dtype: int64
```

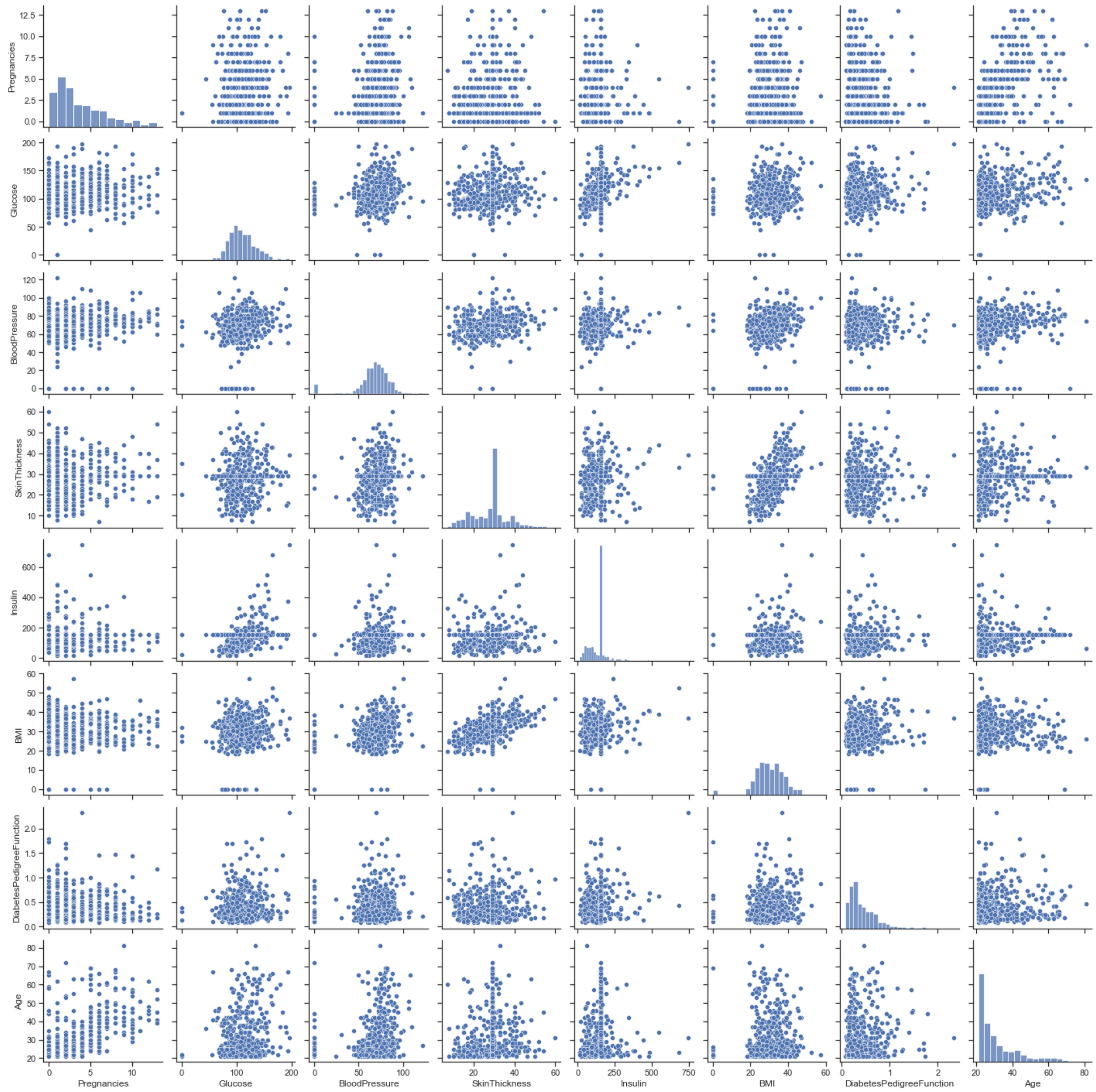
0.1.2 Scatter plots

[253]:
```
#Pair plots for all dataset
sns.set(style="ticks", color_codes=True)
g = sns.pairplot(dataset_imputed,hue="Outcome")
```

[254]:
```
#Pair plots for all Positive cases
sns.set(style="ticks", color_codes=True)
g = sns.
 ↪pairplot(Positive[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI',
 ↪'Age']])
```



[255]:
```
#Pair plots for all Negative cases
sns.set(style="ticks", color_codes=True)
g = sns.
 ↪pairplot(Negative[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI',
 ↪'Age']])
```



0.1.3 Correlation Analysis and Heat map

[256]:
```
### correlation matrix
dataset_imputed.corr()
```

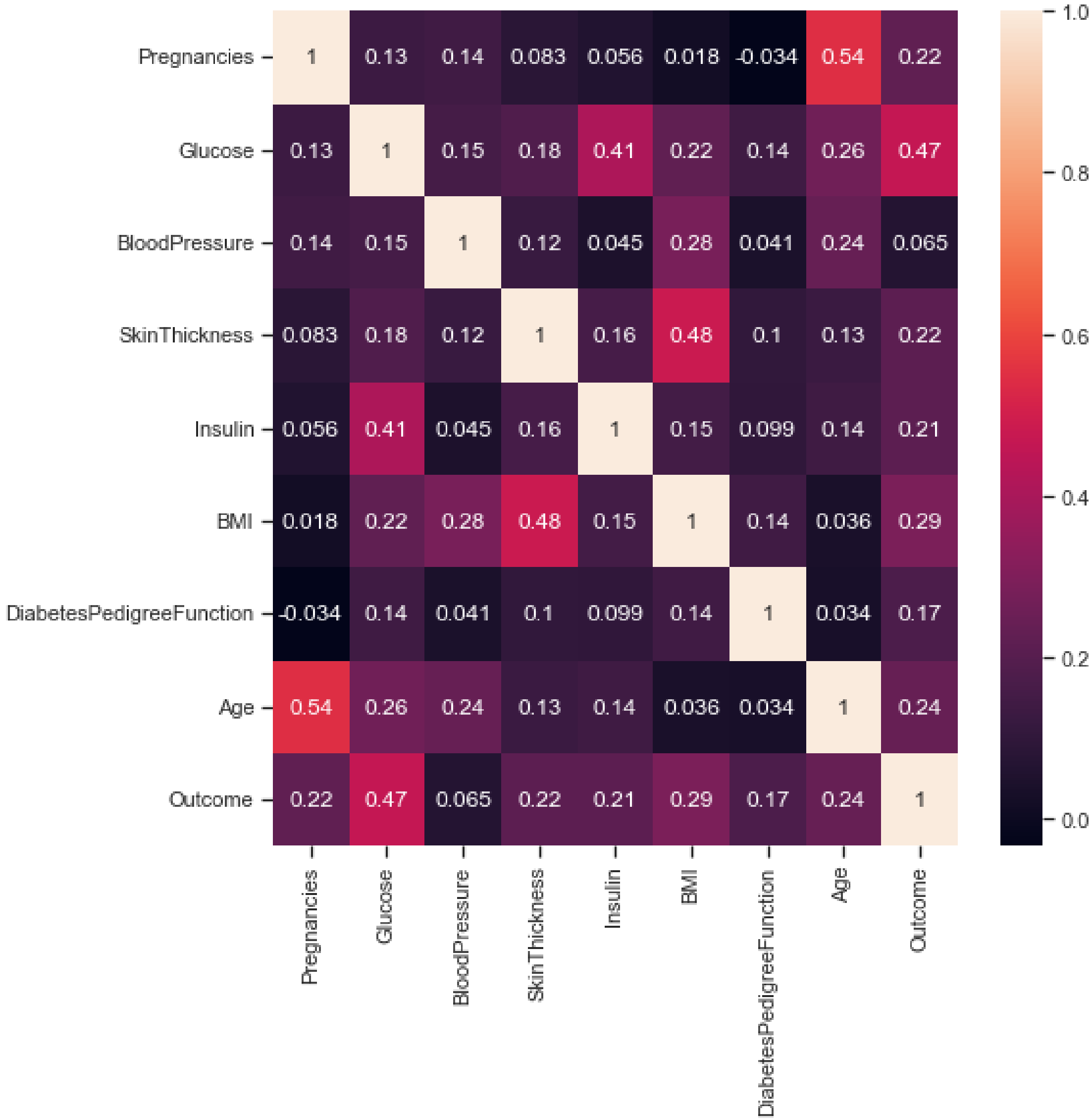
[256]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	\
Pregnancies	1.000000	0.129459	0.141282	0.082989	
Glucose	0.129459	1.000000	0.152590	0.182455	
BloodPressure	0.141282	0.152590	1.000000	0.123444	
SkinThickness	0.082989	0.182455	0.123444	1.000000	
Insulin	0.056027	0.407699	0.045319	0.158139	
BMI	0.017683	0.221071	0.281805	0.480496	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.100966	

```
Age                        0.544341  0.263514       0.239528       0.127872
Outcome                    0.221898  0.466581       0.065068       0.215299

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               0.056027  0.017683                 -0.033523
Glucose                   0.407699  0.221071                  0.137337
BloodPressure             0.045319  0.281805                  0.041265
SkinThickness             0.158139  0.480496                  0.100966
Insulin                   1.000000  0.149468                  0.098634
BMI                       0.149468  1.000000                  0.140647
DiabetesPedigreeFunction  0.098634  0.140647                  1.000000
Age                       0.136734  0.036242                  0.033561
Outcome                   0.214411  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness             0.127872  0.215299
Insulin                   0.136734  0.214411
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

[257]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(dataset_imputed.corr(),annot=True)
```

[257]: <AxesSubplot:>



0.1.4 Project Task: Week 3 and Week 4 – Data Modelling and Model Performance Evaluation

Model 1 : Logistic Regression

[258]:
```
dataset_imputed.head(5)
```

[258]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35.00000	155.548223	33.6	
1	1	85	66	29.00000	155.548223	26.6	
2	8	183	64	29.15342	155.548223	23.3	
3	1	89	66	23.00000	94.000000	28.1	
4	0	137	40	35.00000	168.000000	43.1	

```
   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[259]:
```
features = dataset_imputed.iloc[:,[0,1,2,3,4,5,6,7]].values
label = dataset_imputed.iloc[:,8].values
```

[260]:
```
#Train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(features,
                                                label,
                                                test_size=0.2,
                                                random_state =10)
```

[261]:
```
#Create model
from sklearn.linear_model import LogisticRegression
logRegModel = LogisticRegression()
logRegModel.fit(X_train,y_train)
```

[261]:
```
LogisticRegression()
```

[308]:
```
y_pred = logRegModel.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy of logistic regression classifier on test set',␣
 ↪accuracy_score(y_test, y_pred))
```

```
Accuracy of logistic regression classifier on test set 0.7597402597402597
```

[309]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[86  9]
 [28 31]]
```

Model 2 : Decision Tree Classifier

[310]:
```
#Hyper Parameter tuning of max_dept
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
for i in range(3,20):
    print("For max_depth = ",i)
    DTModel = DecisionTreeClassifier(max_depth=i)
    DTModel.fit(X_train,y_train)
```

```
    y_pred = DTModel.predict(X_test)
    print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
For max_depth =  3
Accuracy: 0.6948051948051948
For max_depth =  4
Accuracy: 0.7532467532467533
For max_depth =  5
Accuracy: 0.7597402597402597
For max_depth =  6
Accuracy: 0.7467532467532467
For max_depth =  7
Accuracy: 0.7597402597402597
For max_depth =  8
Accuracy: 0.7532467532467533
For max_depth =  9
Accuracy: 0.7727272727272727
For max_depth =  10
Accuracy: 0.7727272727272727
For max_depth =  11
Accuracy: 0.7337662337662337
For max_depth =  12
Accuracy: 0.7012987012987013
For max_depth =  13
Accuracy: 0.7012987012987013
For max_depth =  14
Accuracy: 0.7402597402597403
For max_depth =  15
Accuracy: 0.6948051948051948
For max_depth =  16
Accuracy: 0.7142857142857143
For max_depth =  17
Accuracy: 0.6818181818181818
For max_depth =  18
Accuracy: 0.7142857142857143
For max_depth =  19
Accuracy: 0.7272727272727273
```

Highest Accuracy of Decision Tree Model can be obtained on Max_Depth = 10

[311]:
```
DTModel = DecisionTreeClassifier(max_depth=10)
DTModel.fit(X_train,y_train)
y_pred = DTModel.predict(X_test)
```

[312]:
```
DTModel.score(X_train,y_train)
```

[312]:
```
0.9267100977198697
```

[313]:
```
DTModel.score(X_test,y_test)
```

[313]: 0.7532467532467533

[315]:
```
print('Accuracy of Decision Tree regression classifier on test set',␣
↪accuracy_score(y_test, y_pred))
```

```
Accuracy of Decision Tree regression classifier on test set 0.7532467532467533
```

[272]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[77 18]
 [20 39]]
```

Model 3 : Random Forest Classifier

[277]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

[281]:
```
rfModel = RandomForestClassifier(n_estimators=60)
rfModel.fit(X_train, y_train)
y_pred = rfModel.predict(X_test)
```

[317]:
```
print('Accuracy of Random Forest regression classifier on test set',␣
↪accuracy_score(y_test, y_pred))
```

```
Accuracy of Random Forest regression classifier on test set 0.7532467532467533
```

[286]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[85 10]
 [27 32]]
```

Model 4 : Support Vector Machine

[291]:
```
#Support Vector Classifier

from sklearn.svm import SVC
SVMmodel = SVC(kernel='rbf',
           gamma='auto')
SVMmodel.fit(X_train,y_train)
```

[291]: SVC(gamma='auto')

[318]:
```
y_pred=SVMmodel.predict(X_test)
```

[319]:
```
print('Accuracy of Support Vector Machine on test set', accuracy_score(y_test,␣
↪y_pred))
```

```
Accuracy of Support Vector Machine on test set 0.6168831168831169
```

Model 5 : KNN Classifier

[294]:
```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
knnClassifier = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
knnClassifier.fit(X_train,y_train)
```

[294]:
```
KNeighborsClassifier(n_neighbors=7)
```

[320]:
```
y_pred=knnClassifier.predict(X_test)
```

[321]:
```
print('Accuracy of KNN Classifier on test set', accuracy_score(y_test, y_pred))
```

```
Accuracy of KNN Classifier on test set 0.7272727272727273
```

We observed that Random Forest is best performing model for this dataset

Accuracy of 75%

[ ]:
```

```