Neeti Capstone project 2 (Healthcare)

April 8, 2022

```
[1]: import pandas as pd
```

```
[2]: data=pd.read_csv("health care diabetes.csv")
     df=pd.read_csv("healthcare appointment data.csv")
     df_train=pd.read_csv("train.csv")
```

```
[3]: data
```

```
[3]:      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
     ..    …   …     …      …  …  …
     763           10      101             76             48      180  32.9
     764            2      122             70             27        0  36.8
     765            5      121             72             23      112  26.2
     766            1      126             60              0        0  30.1
     767            1       93             70             31        0  30.4

          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
     ..           … …  …
     763                     0.171   63        0
     764                     0.340   27        0
     765                     0.245   30        0
     766                     0.349   47        1
     767                     0.315   23        0

     [768 rows x 9 columns]
```

```
[4]: type(data)
```

[4]:
```
pandas.core.frame.DataFrame
```

[5]:
```
data.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    int64  
 4   Insulin                   768 non-null    int64  
 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(2), int64(7)
memory usage: 54.1 KB
```

[6]:
```
data.describe()
```

[6]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000   
mean      3.845052  120.894531      69.105469      20.536458   79.799479   
std       3.369578   31.972618      19.355807      15.952218  115.244002   
min       0.000000    0.000000       0.000000       0.000000    0.000000   
25%       1.000000   99.000000      62.000000       0.000000    0.000000   
50%       3.000000  117.000000      72.000000      23.000000   30.500000   
75%       6.000000  140.250000      80.000000      32.000000  127.250000   
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  
```

[7]:
```
data.isnull()
```

[7]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0          False    False          False          False    False  False   
```

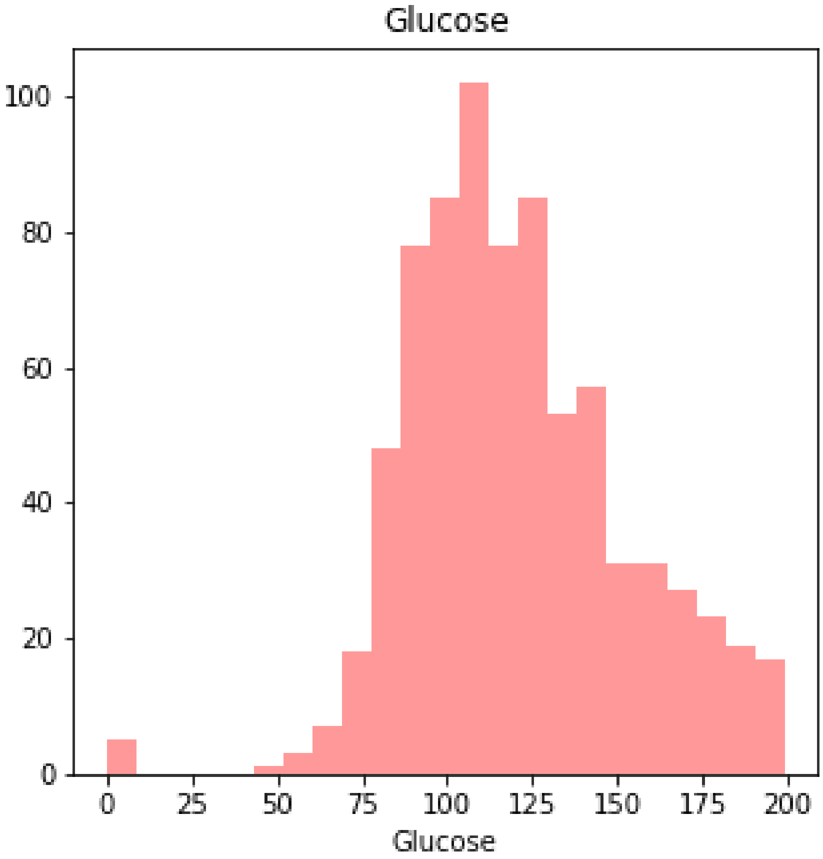
```
1          False    False          False          False    False  False
2          False    False          False          False    False  False
3          False    False          False          False    False  False
4          False    False          False          False    False  False
..           …      …            …            …      …    …
763        False    False          False          False    False  False
764        False    False          False          False    False  False
765        False    False          False          False    False  False
766        False    False          False          False    False  False
767        False    False          False          False    False  False

     DiabetesPedigreeFunction    Age  Outcome
0                       False  False    False
1                       False  False    False
2                       False  False    False
3                       False  False    False
4                       False  False    False
..                        …    …      …
763                     False  False    False
764                     False  False    False
765                     False  False    False
766                     False  False    False
767                     False  False    False

[768 rows x 9 columns]
```

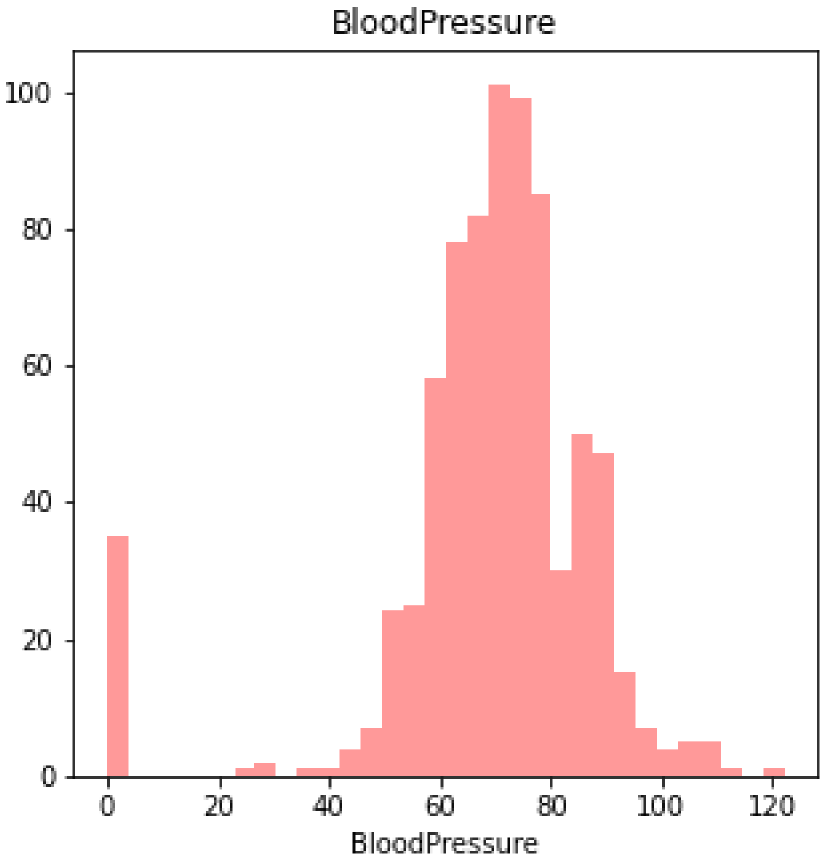
[8]:
```
miss_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin'
, 'BMI']
```

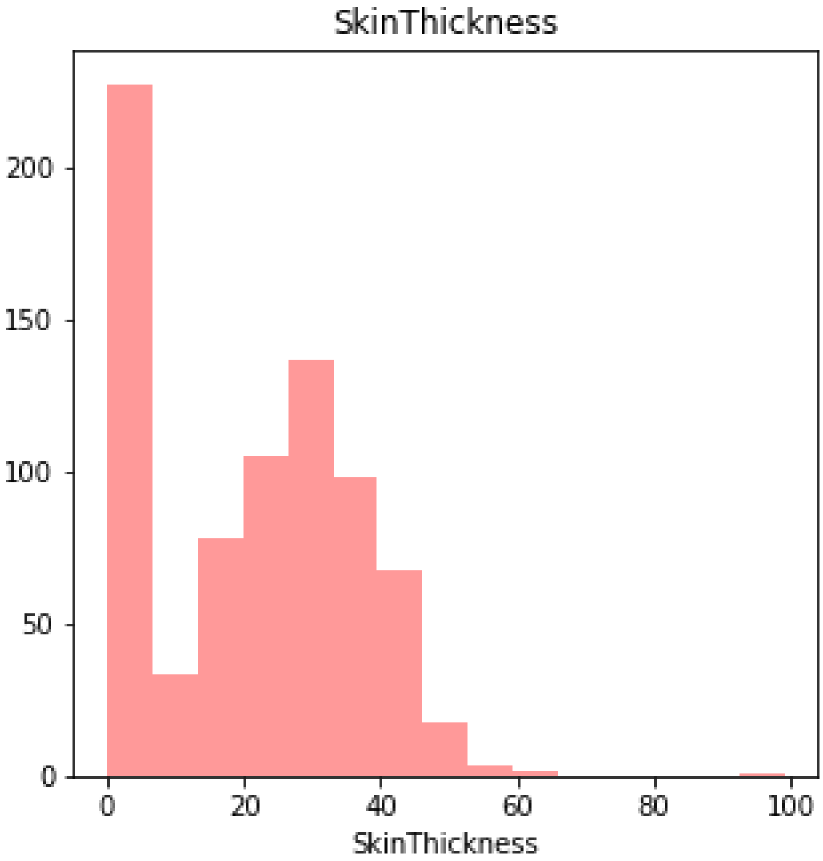
[9]:
```
#Importing libraries

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

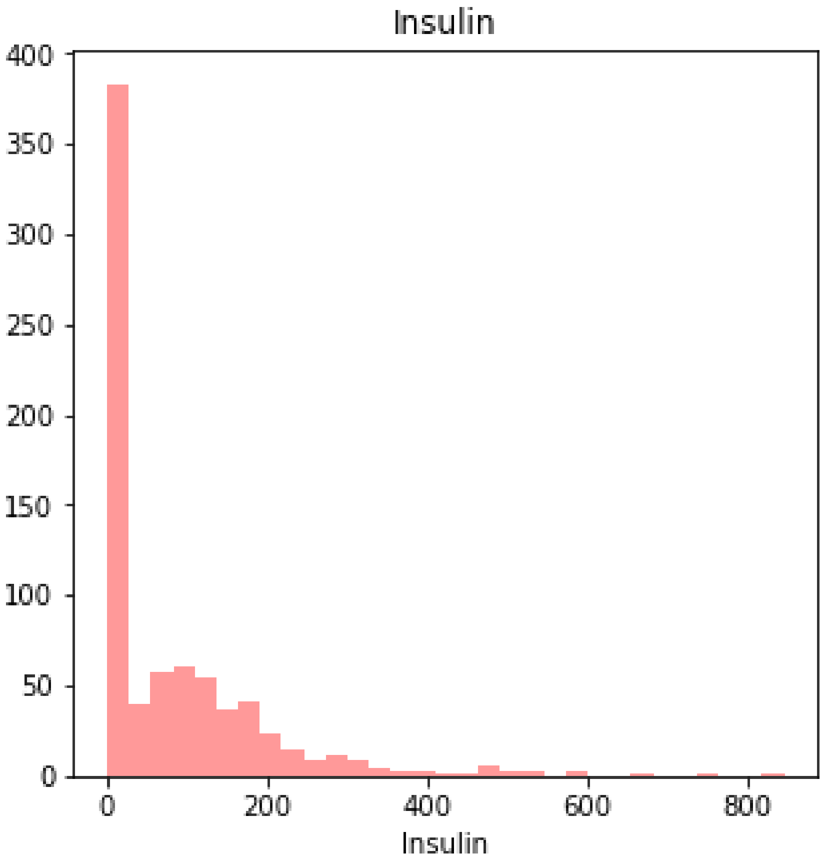
[10]:
```
for col in miss_cols:
    plt.figure(figsize = (5, 5))
    plt.title(col)
    sns.distplot(data[col], kde = False, color = 'red')
```

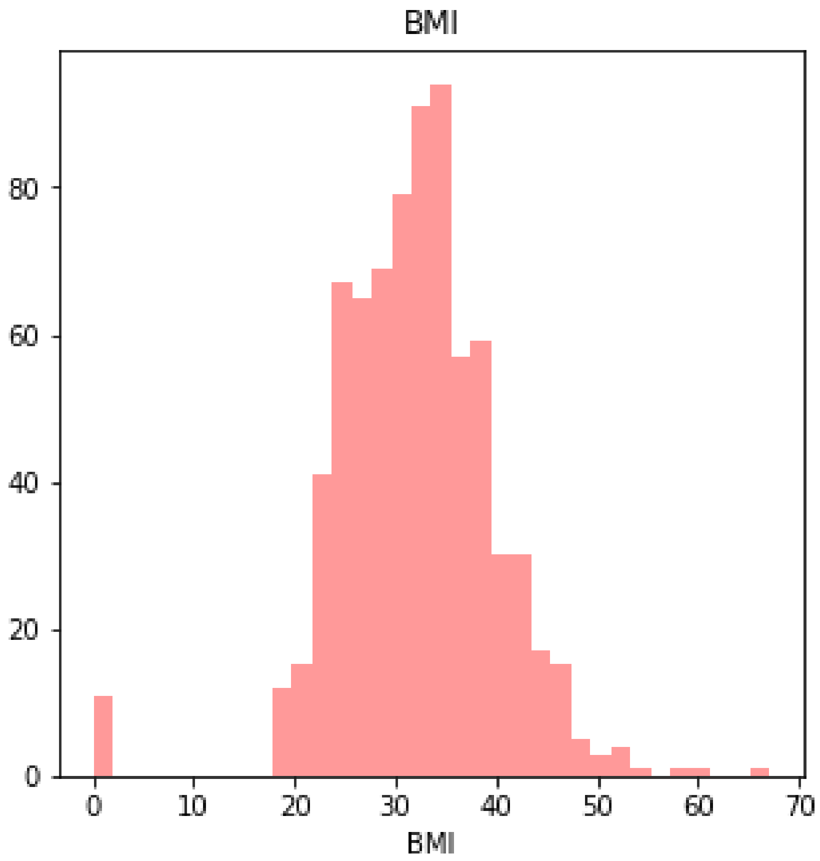
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```


Glucose
100
80
60
40
20
0
0
25
50
75
100
125
150
175
200
Glucose


BloodPressure
100
80
60
40
20
0
0
20
40
60
80
100
120
BloodPressure

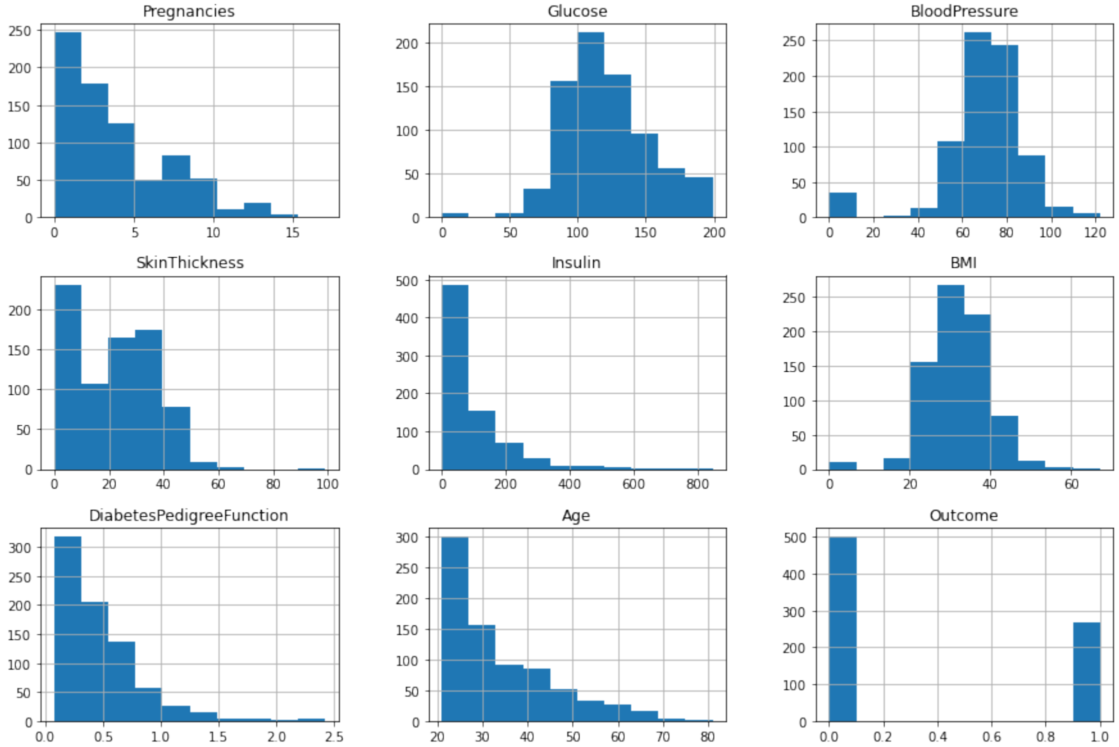

SkinThickness
200
150
100
50
0
0
20
40
60
80
100
SkinThickness


Insulin
400
350
300
250
200
150
100
50
0
0
200
400
600
800
Insulin



[11]:

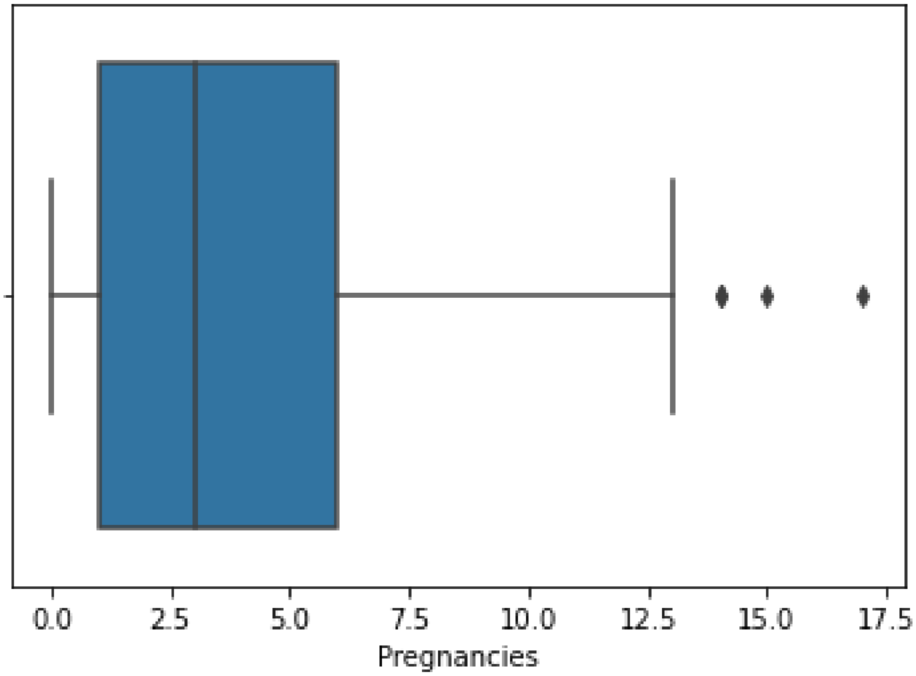
```
data.hist(figsize=(15,10))
plt.show()
```



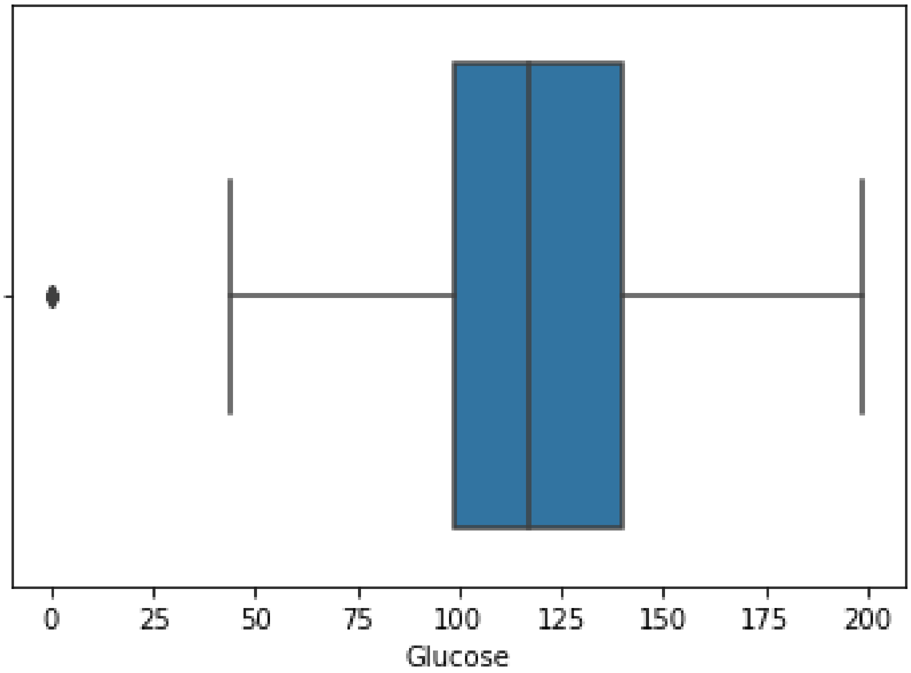
[12]:

```
for column in data.columns:
 sns.boxplot(data[column])
 plt.show()
```

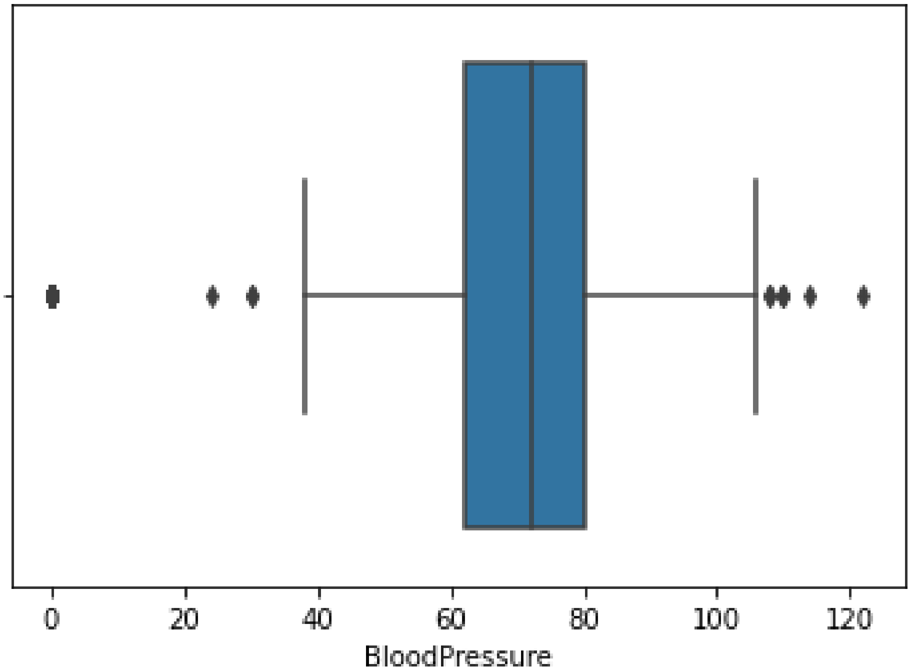
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



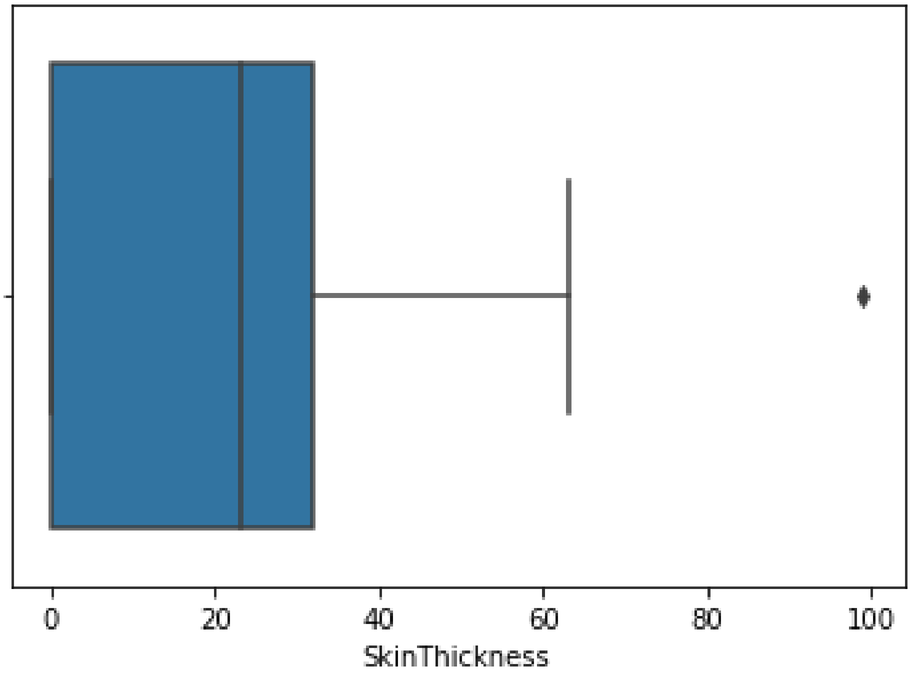
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



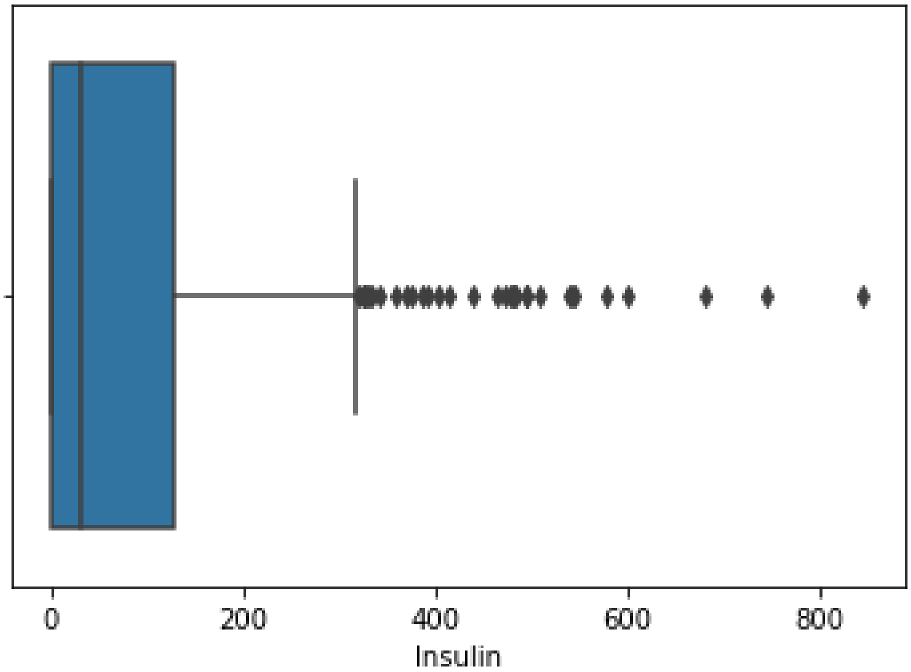
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



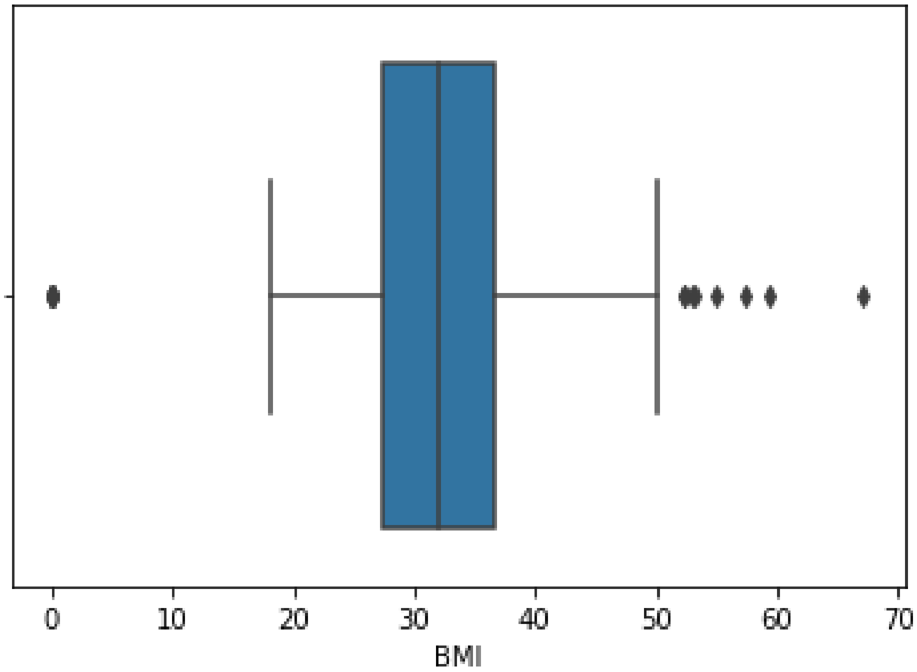
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



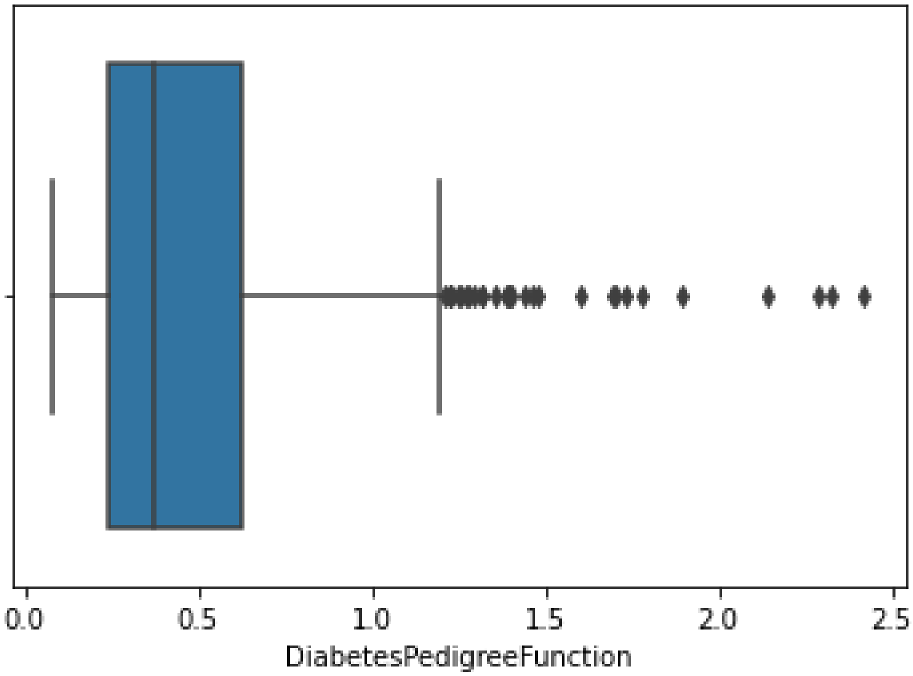
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



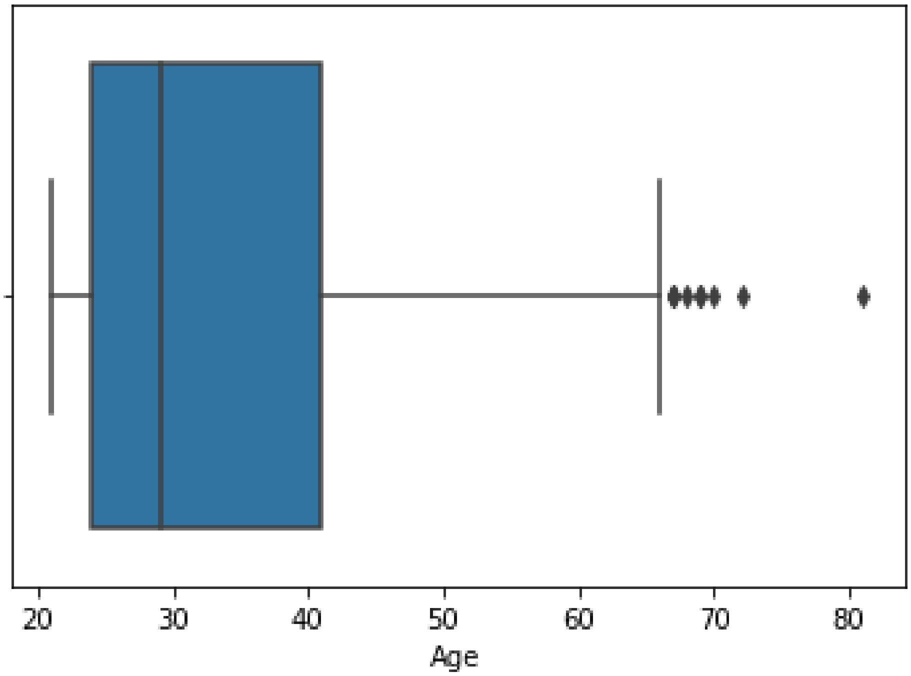
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



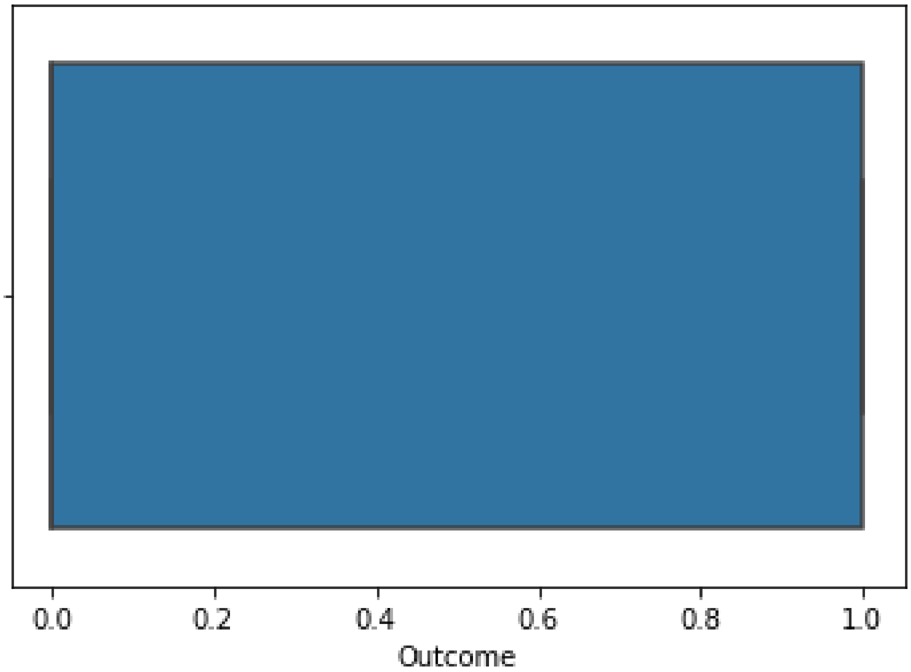
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[ ]:

```
[13]: data[['Glucose', 'BloodPressure', 'SkinThickness',
      'Insulin', 'BMI']] = data[['Glucose', 'BloodPressure',
      'SkinThickness', 'Insulin', 'BMI']].replace(0, np.nan)
```

```
[14]: data
```

```
[14]:      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
      0              6    148.0           72.0           35.0      NaN  33.6
      1              1     85.0           66.0           29.0      NaN  26.6
      2              8    183.0           64.0            NaN      NaN  23.3
      3              1     89.0           66.0           23.0     94.0  28.1
      4              0    137.0           40.0           35.0    168.0  43.1
      ..           …        …            …             …        …     …
      763           10    101.0           76.0           48.0    180.0  32.9
      764            2    122.0           70.0           27.0      NaN  36.8
      765            5    121.0           72.0           23.0    112.0  26.2
      766            1    126.0           60.0            NaN      NaN  30.1
      767            1     93.0           70.0           31.0      NaN  30.4

           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
..                         …   …       …
763                      0.171   63         0
764                      0.340   27         0
765                      0.245   30         0
766                      0.349   47         1
767                      0.315   23         0

[768 rows x 9 columns]
```

```
[15]: pip install impyute
```

```
Requirement already satisfied: impyute in c:\users\91820\anaconda3\lib\site-
packages (0.0.8)
Requirement already satisfied: scikit-learn in
c:\users\91820\anaconda3\lib\site-packages (from impyute) (1.0.2)
Requirement already satisfied: numpy in c:\users\91820\anaconda3\lib\site-
packages (from impyute) (1.20.3)
Requirement already satisfied: scipy in c:\users\91820\anaconda3\lib\site-
packages (from impyute) (1.7.1)
Requirement already satisfied: joblib>=0.11 in
c:\users\91820\anaconda3\lib\site-packages (from scikit-learn->impyute) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\91820\anaconda3\lib\site-packages (from scikit-learn->impyute) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[16]: from impyute.imputation.cs import fast_knn
```

```
[17]: imputed_data = fast_knn(data.values, k = 30)
```

```
[18]: imputed_data
```

```
[18]: array([[  6.   , 148.   ,  72.   , …,   0.627,  50.   ,   1.   ],
             [  1.   ,  85.   ,  66.   , …,   0.351,  31.   ,   0.   ],
             [  8.   , 183.   ,  64.   , …,   0.672,  32.   ,   1.   ],
             …,
             [  5.   , 121.   ,  72.   , …,   0.245,  30.   ,   0.   ],
             [  1.   , 126.   ,  60.   , …,   0.349,  47.   ,   1.   ],
             [  1.   ,  93.   ,  70.   , …,   0.315,  23.   ,   0.   ]])
```

```
[19]: imputed_data = pd.DataFrame(imputed_data)
```

```
[20]: imputed_data.head()
```

```
[20]:      0      1     2          3           4     5      6     7    8
      0  6.0  148.0  72.0  35.000000  155.333764  33.6  0.627  50.0  1.0
```

```
1  1.0   85.0  66.0  29.000000  155.548223  26.6  0.351  31.0  0.0
2  8.0  183.0  64.0  29.367818  155.374337  23.3  0.672  32.0  1.0
3  1.0   89.0  66.0  23.000000   94.000000  28.1  0.167  21.0  0.0
4  0.0  137.0  40.0  35.000000  168.000000  43.1  2.288  33.0  1.0
```

[21]:
```
data.head()
```

[21]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6    148.0           72.0           35.0      NaN  33.6
1            1     85.0           66.0           29.0      NaN  26.6
2            8    183.0           64.0            NaN      NaN  23.3
3            1     89.0           66.0           23.0     94.0  28.1
4            0    137.0           40.0           35.0    168.0  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
```

[22]:
```
data.columns
```

[22]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[23]:
```
imputed_data.columns = ['Pregnancies', 'Glucose', 'BloodPressure',␣
 ↪'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

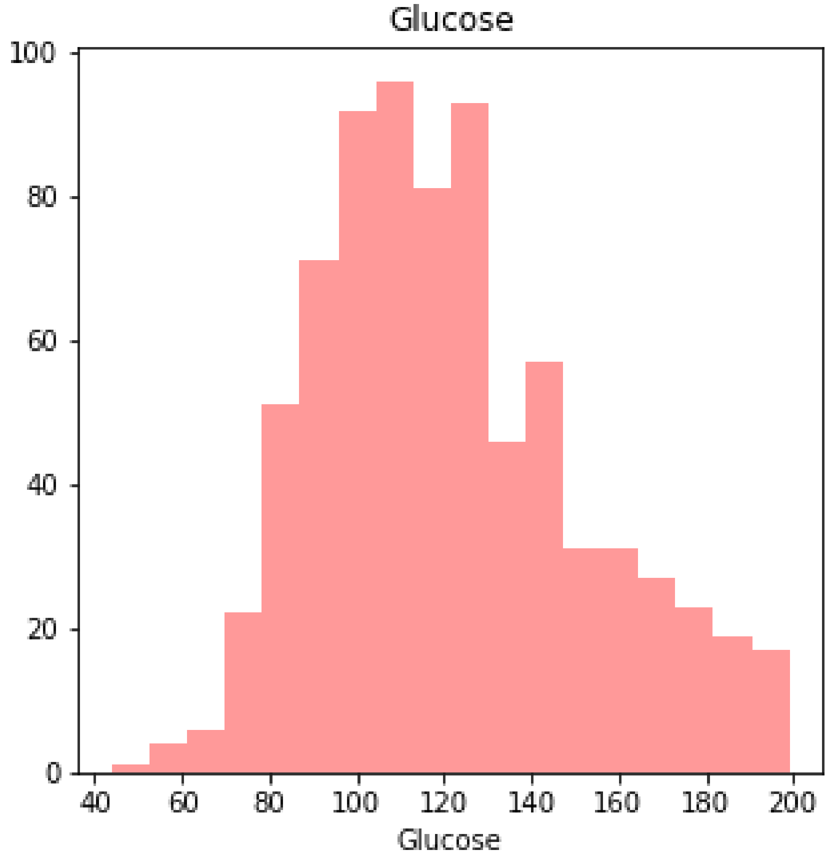
[24]:
```
imputed_data.head()
```

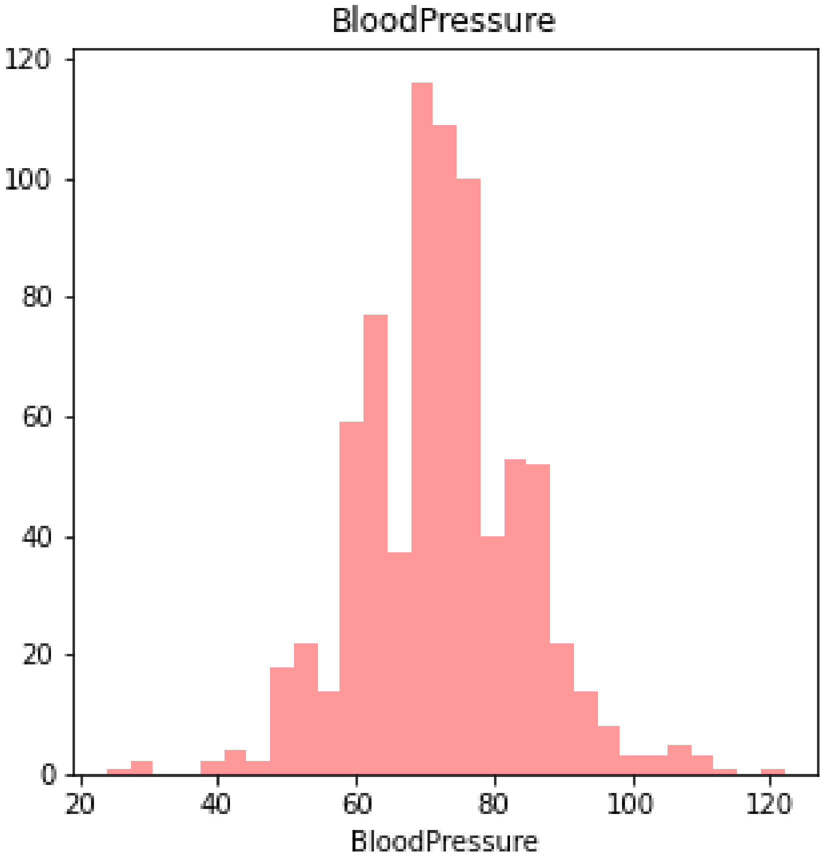
[24]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0          6.0    148.0           72.0      35.000000  155.333764  33.6
1          1.0     85.0           66.0      29.000000  155.548223  26.6
2          8.0    183.0           64.0      29.367818  155.374337  23.3
3          1.0     89.0           66.0      23.000000   94.000000  28.1
4          0.0    137.0           40.0      35.000000  168.000000  43.1

   DiabetesPedigreeFunction   Age  Outcome
0                     0.627  50.0      1.0
1                     0.351  31.0      0.0
2                     0.672  32.0      1.0
3                     0.167  21.0      0.0
4                     2.288  33.0      1.0
```

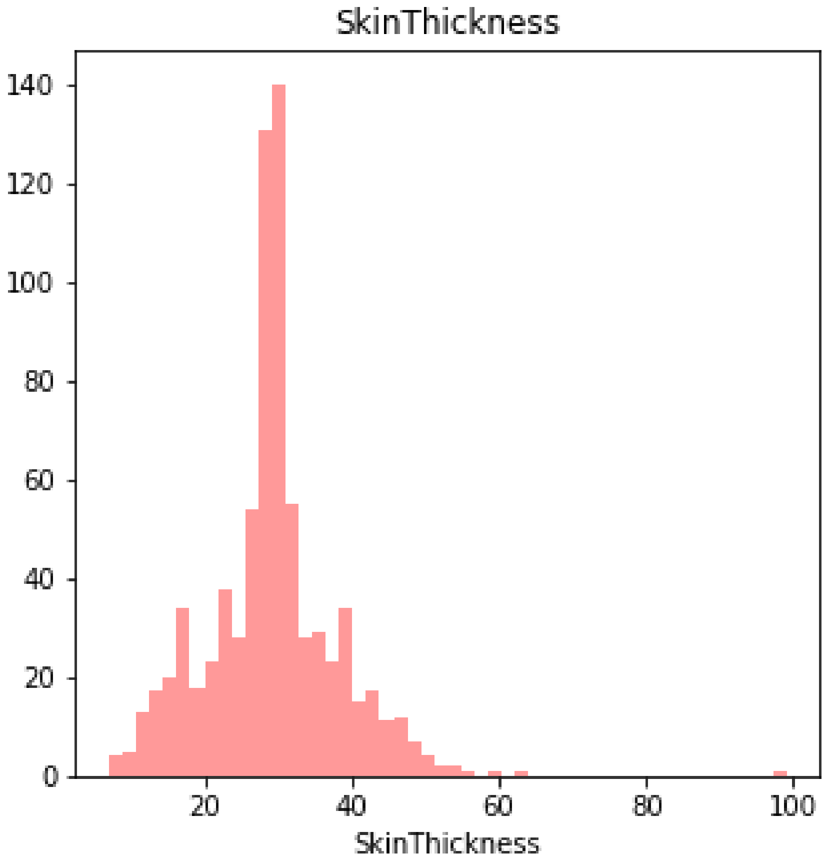
[25]:

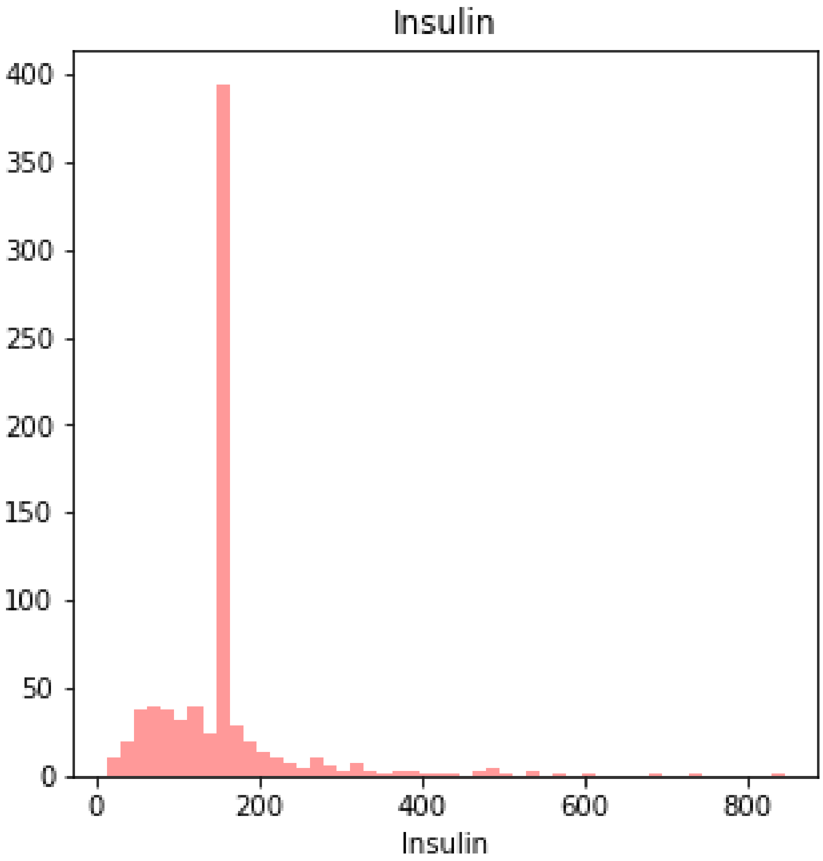
```
for col in miss_cols:
    plt.figure(figsize = (5, 5))
    plt.title(col)
    sns.distplot(imputed_data[col], kde = False, color = 'red')
```

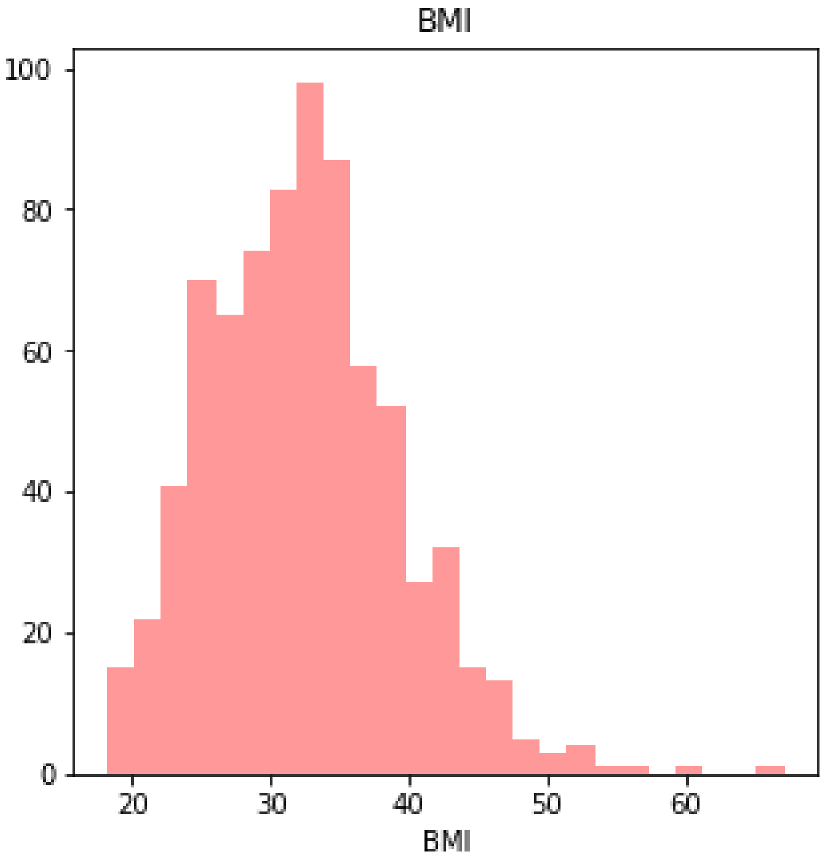
```
C:\Users\91820\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```




BloodPressure
120
100
80
60
40
20
0
20
40
60
80
100
120
BloodPressure


SkinThickness
140
120
100
80
60
40
20
0
20
40
60
80
100
SkinThickness


Insulin
400
350
300
250
200
150
100
50
0
0
200
400
600
800
Insulin



[26]:

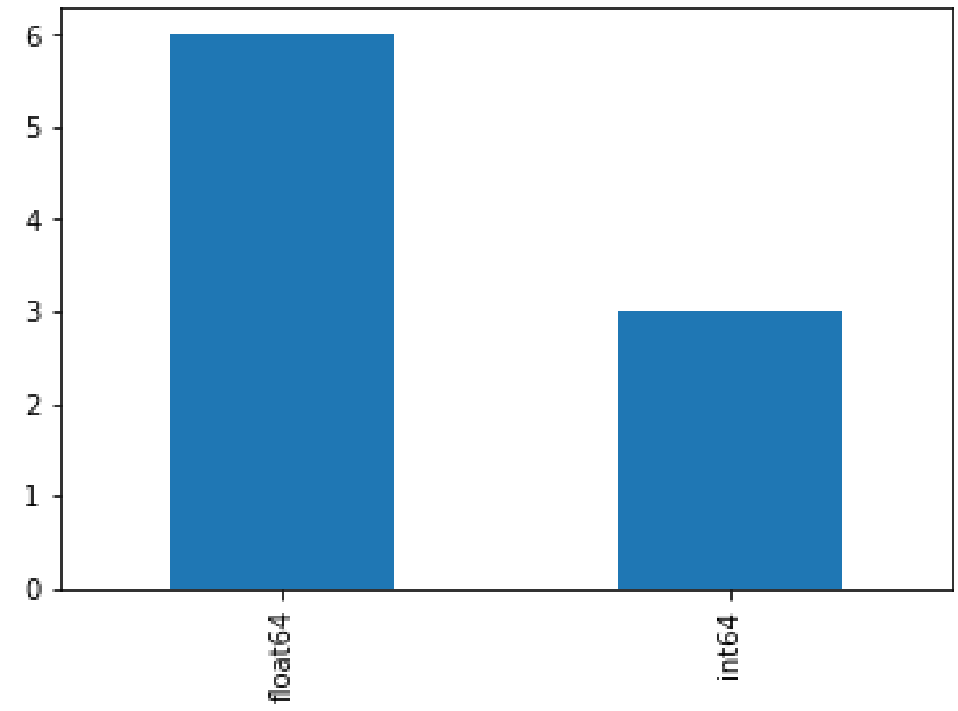
```
imputed_data.dtypes
```

[26]:

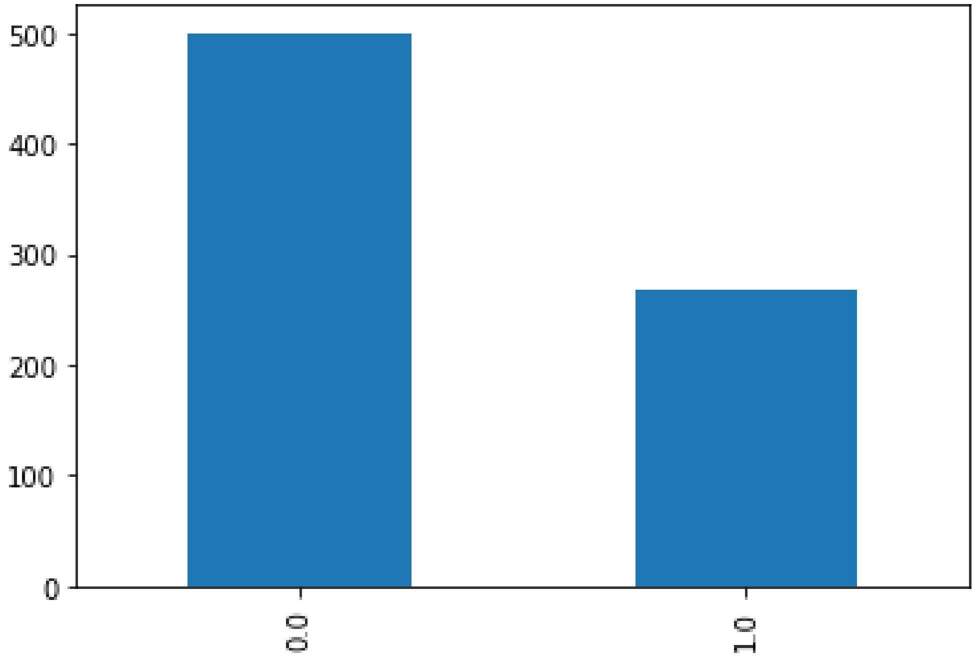
```
Pregnancies                 float64
Glucose                     float64
BloodPressure               float64
SkinThickness               float64
Insulin                     float64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                         float64
Outcome                     float64
dtype: object
```

[27]:

```
(data.dtypes).value_counts().plot(kind = 'bar')
plt.show()
```



[28]:
```
(imputed_data.Outcome).value_counts().plot(kind = 'bar')
plt.show()
```



[29]:

```
 imputed_data.Outcome.value_counts()
```

[29]:

```
0.0    500
1.0    268
Name: Outcome, dtype: int64
```

[30]:

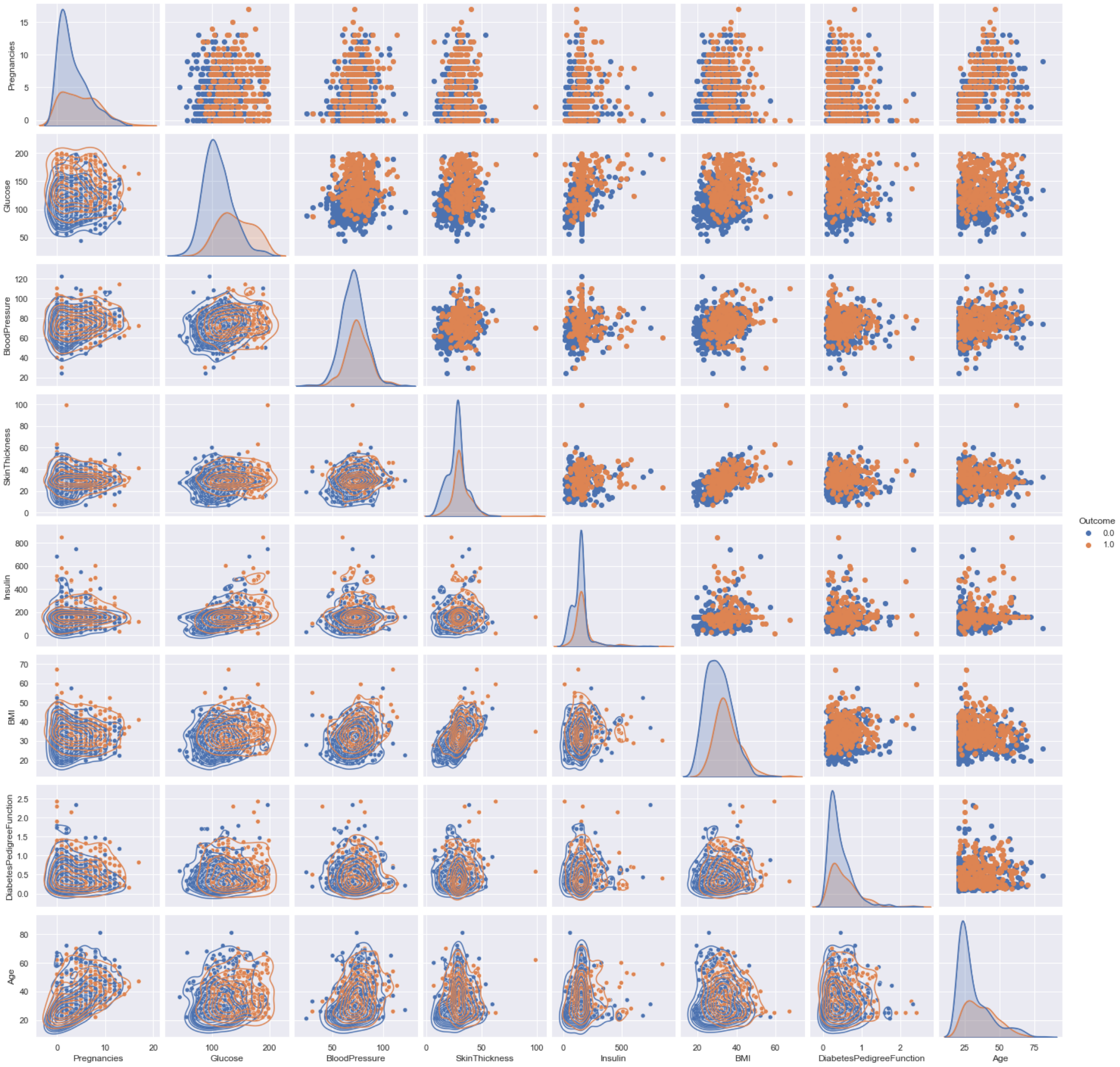
```
round(imputed_data.Outcome.value_counts(normalize = True)*100, 2)
```

[30]:

```
0.0    65.1
1.0    34.9
Name: Outcome, dtype: float64
```

[31]:

```
sns.set()
g = sns.pairplot(imputed_data, hue = 'Outcome')
g.map_lower(sns.kdeplot)
g.map_upper(plt.scatter)
g.map_diag(sns.kdeplot)
plt.show()
```



```
[32]: round(imputed_data.corr()['Outcome'][:], 3).sort_values(ascending = False)
```

```
[32]: Outcome                     1.000
      Glucose                     0.494
      BMI                         0.314
      Age                         0.238
      SkinThickness               0.226
      Pregnancies                 0.222
      Insulin                     0.214
      DiabetesPedigreeFunction    0.174
      BloodPressure               0.171
      Name: Outcome, dtype: float64
```

[33]:

```
def color_negative_red(value):
       """ Colors elements in a dataframe green if positive and red if␣
↪negative. Does not color NaN values."""
       if value < -0.1:
           color = 'red'
       elif value > 0.1:
           color = 'green'
       else:
           color = 'white'
       return 'color: %s' % color
```

[34]:

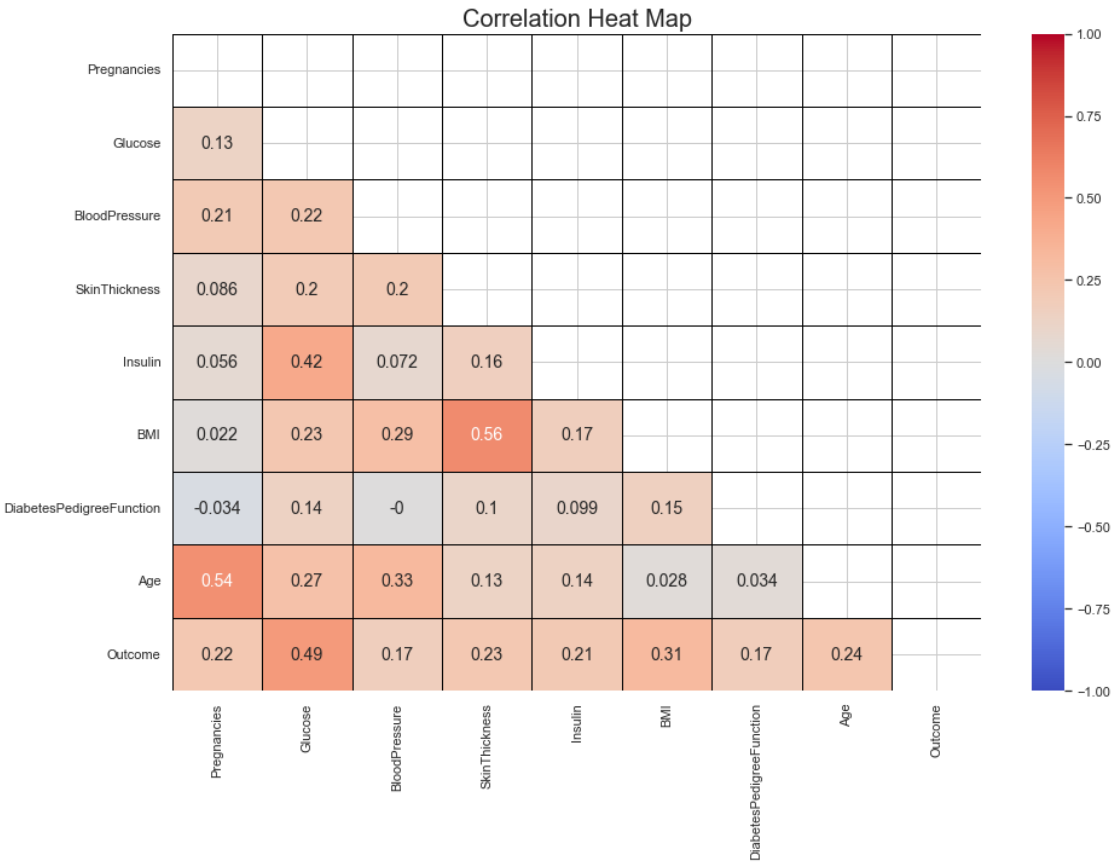
```
round(imputed_data.corr(), 3).style.applymap(color_negative_red)
```

[34]: <pandas.io.formats.style.Styler at 0x196ea272ca0>

[35]:

```
 sns.set_style("whitegrid")
corr = data.corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
#kot = corr[corr>=.6]
plt.figure(figsize=(15,10))
sns.heatmap(round(imputed_data.corr(), 3), cmap="coolwarm", vmin=-1,
vmax=1, annot = True, mask = mask, linewidths=1, linecolor='black',
annot_kws={"fontsize":14}).set_title('Correlation Heat Map', fontsize = 20)
plt.grid('on', )
plt.show()
```

```
C:\Users\91820\AppData\Local\Temp/ipykernel_32068/4172546160.py:3:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To
silence this warning, use `bool` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(corr, dtype=np.bool)
```



[36]:
```
# strong correlation seen between:
   #Age & Pregnancies , Glucose & Outcome , BMI & Skin thickness
```

DATA MODELLING

[37]:
```
imputed_data.head()
```

[37]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0          6.0    148.0           72.0      35.000000  155.333764  33.6
1          1.0     85.0           66.0      29.000000  155.548223  26.6
2          8.0    183.0           64.0      29.367818  155.374337  23.3
3          1.0     89.0           66.0      23.000000   94.000000  28.1
4          0.0    137.0           40.0      35.000000  168.000000  43.1

   DiabetesPedigreeFunction   Age  Outcome
0                     0.627  50.0      1.0
1                     0.351  31.0      0.0
2                     0.672  32.0      1.0
3                     0.167  21.0      0.0
```

```
4                      2.288  33.0      1.0
```

[38]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

[39]:
```
imputed_data_scaled = scaler.fit_transform(imputed_data)
```

[40]:
```
 imputed_data_scaled = pd.DataFrame(imputed_data_scaled, columns=imputed_data.
 ↪columns)
```

[41]:
```
imputed_data_scaled.head()
```

[41]:
```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
0     0.352941  0.670968       0.489796       0.304348  0.169872  0.314928
1     0.058824  0.264516       0.428571       0.239130  0.170130  0.171779
2     0.470588  0.896774       0.408163       0.243128  0.169921  0.104294
3     0.058824  0.290323       0.428571       0.173913  0.096154  0.202454
4     0.000000  0.600000       0.163265       0.304348  0.185096  0.509202

   DiabetesPedigreeFunction       Age  Outcome
0                  0.234415  0.483333      1.0
1                  0.116567  0.166667      0.0
2                  0.253629  0.183333      1.0
3                  0.038002  0.000000      0.0
4                  0.943638  0.200000      1.0
```

[42]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

[43]:
```
imputed_data_scaled.head()
```

[43]:
```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
0     0.352941  0.670968       0.489796       0.304348  0.169872  0.314928
1     0.058824  0.264516       0.428571       0.239130  0.170130  0.171779
2     0.470588  0.896774       0.408163       0.243128  0.169921  0.104294
3     0.058824  0.290323       0.428571       0.173913  0.096154  0.202454
4     0.000000  0.600000       0.163265       0.304348  0.185096  0.509202

   DiabetesPedigreeFunction       Age  Outcome
0                  0.234415  0.483333      1.0
1                  0.116567  0.166667      0.0
2                  0.253629  0.183333      1.0
3                  0.038002  0.000000      0.0
4                  0.943638  0.200000      1.0
```

[44]:

```
y = imputed_data_scaled['Outcome']
x = imputed_data_scaled.drop('Outcome', axis = 1)
```

[45]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0.2,␣
↪stratify = y)
```

[46]:

```
x_train.head()
```

[46]:

```
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
542     0.588235  0.296774       0.622449       0.271739  0.170130  0.341513
646     0.058824  0.793548       0.510204       0.108696  0.156250  0.106339
525     0.176471  0.277419       0.367347       0.119565  0.170355  0.073620
386     0.294118  0.464516       0.510204       0.239130  0.170130  0.288344
157     0.058824  0.419355       0.326531       0.152174  0.145433  0.143149

     DiabetesPedigreeFunction       Age
542                  0.318958  0.583333
646                  0.157558  0.200000
525                  0.156277  0.000000
386                  0.248506  0.233333
157                  0.322374  0.033333
```

[47]:

```
y_train.head()
```

[47]:

```
542    1.0
646    1.0
525    0.0
386    1.0
157    0.0
Name: Outcome, dtype: float64
```

[48]:

```
lr=LogisticRegression()
```

[49]:

```
lr.fit(x_train, y_train)
```

[49]:

```
LogisticRegression()
```

[50]:

```
pred = lr.predict(x_test)
```

[51]:

```
cnf_matrix = confusion_matrix(y_test, pred)
```

[52]:

```
cnf_matrix
```

[52]:

```
array([[89, 11],
       [26, 28]], dtype=int64)
```

[53]:

```
def sens_spec(cnf_matrix):
 total_cm = sum(sum(cnf_matrix))
 accuracy_clf = (cnf_matrix[0,0] + cnf_matrix[1,1]) / total_cm
 sensitivity_clf = cnf_matrix[0,0] / (cnf_matrix[0, 0] + cnf_matrix[0, 1])
 specificity_clf = cnf_matrix[1,1] / (cnf_matrix[1, 0] + cnf_matrix[1, 1])

 #print('accuracy of {} is {}'.format(accuracy)
 return('Accuracy: {}'.format(round(accuracy_clf, 2)), 'Sensitivity: {}'.
 ↪format(round(sensitivity_clf, 2)), 'Specificity: {}'.
 ↪format(round(specificity_clf, 2)))
```

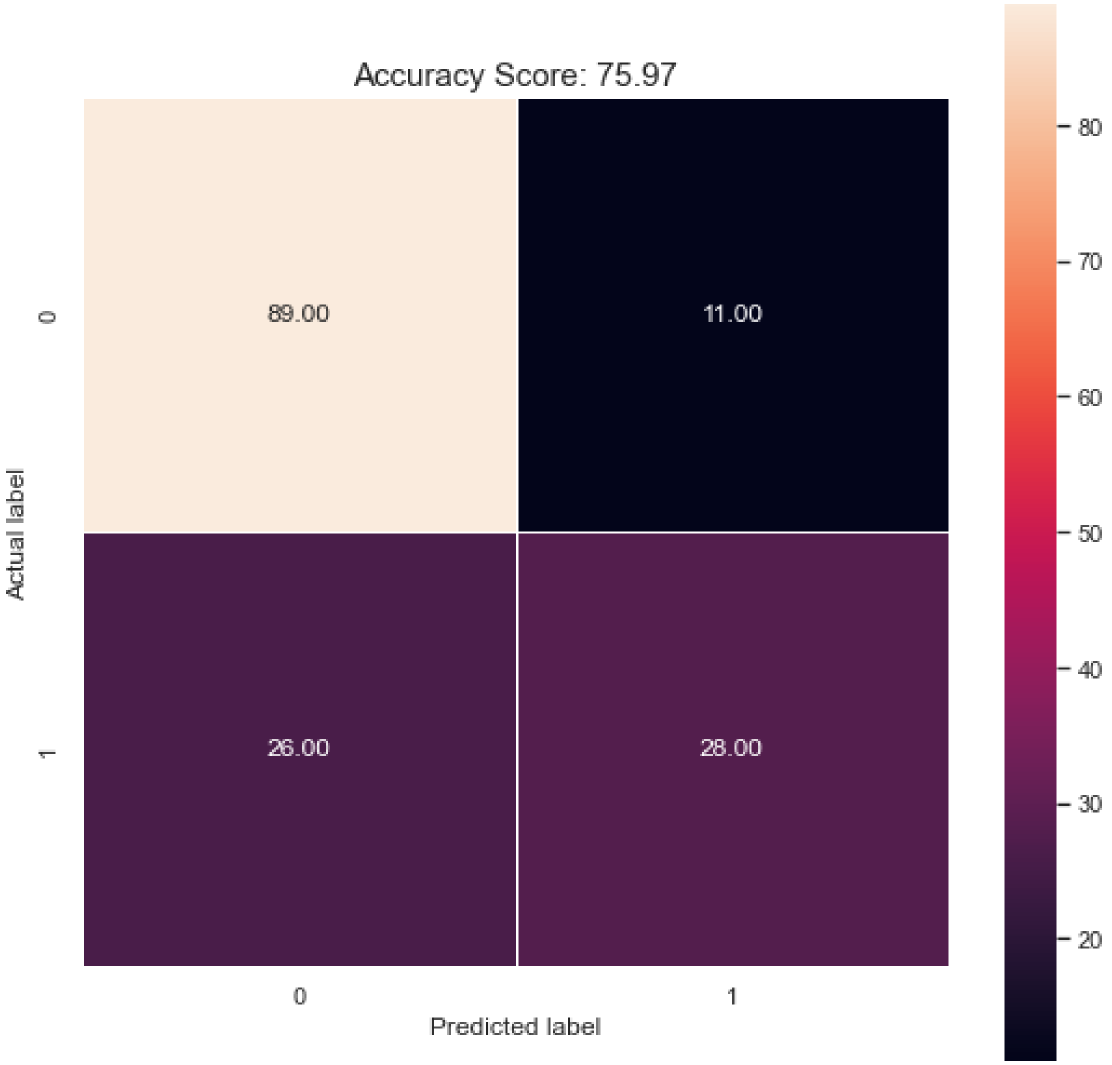
[54]:

```
 # Use score method to get accuracy of model
score = lr.score(x_test, y_test)
print(score)
```

```
0.7597402597402597
```

[55]:

```
plt.figure(figsize=(9,9))
sns.heatmap(cnf_matrix, annot=True, fmt=".2f", linewidths=.5, square = True);
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(round(score*100, 2
))
plt.title(all_sample_title, size = 15);
```



```
[56]: print(sens_spec(cnf_matrix))
      print()
      print(classification_report(y_test, pred))
```

```
('Accuracy: 0.76', 'Sensitivity: 0.89', 'Specificity: 0.52')

              precision    recall  f1-score   support

         0.0       0.77      0.89      0.83       100
         1.0       0.72      0.52      0.60        54

    accuracy                           0.76       154
   macro avg       0.75      0.70      0.72       154
weighted avg       0.75      0.76      0.75       154
```

[57]:

```
from imblearn.over_sampling import SMOTE
```

[58]:

```
os = SMOTE(random_state=0)
```

[59]:

```
columns = x_train.columns
```

[60]:

```
os_data_X,os_data_y=os.fit_resample(x, y)
os_data_X = pd.DataFrame(data=os_data_X,columns=columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=['Outcome'])
# we can Check the numbers of our data
print("length of oversampled data is ",len(os_data_X))
print("Number of NEGATIVE in oversampled␣
 ↪data",len(os_data_y[os_data_y['Outcome']==0]))
print("Number of POSITIVE",len(os_data_y[os_data_y['Outcome']==1]))
print("Proportion of NEGATIVE data in oversampled data is␣
 ↪",len(os_data_y[os_data_y['Outcome']==0])/len(os_data_X))
print("Proportion of POSITIVE data in oversampled data is␣
 ↪",len(os_data_y[os_data_y['Outcome']==1])/len(os_data_X))
```

```
length of oversampled data is  1000
Number of NEGATIVE in oversampled data 500
Number of POSITIVE 500
Proportion of NEGATIVE data in oversampled data is  0.5
Proportion of POSITIVE data in oversampled data is  0.5
```

[61]:

```
X_train, X_test, y_train, y_test = train_test_split(os_data_X,os_data_y,␣
 ↪test_size=0.2, random_state=0)
```

[62]:

```
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

[65]:

```
import scikitplot as skplt
```

```
            ␣
↪---------------------------------------------------------------------------

      ModuleNotFoundError                       Traceback (most recent call␣
↪last)
```

```
    ~\AppData\Local\Temp/ipykernel_32068/105772748.py in <module>
----> 1 import scikitplot as skplt


    ModuleNotFoundError: No module named 'scikitplot'
```

[66]:

```
names = ["Nearest Neighbors", "Logistic Regression", "Linear SVM",
"RBF SVM", "Gaussian Process",
"Decision Tree", "Random Forest", "Neural Net", "AdaBoost"
,
"Naive Bayes", "QDA"]
classifiers = [
KNeighborsClassifier(3),
LogisticRegression(),
SVC(kernel="linear", C=0.025, probability=True),
SVC(gamma=2, C=1, probability=True),
GaussianProcessClassifier(1.0 * RBF(1.0)),
DecisionTreeClassifier(max_depth=5),
RandomForestClassifier(max_depth=5, n_estimators=10,␣
 ↪max_features=1),MLPClassifier(alpha=1, max_iter=1000),
AdaBoostClassifier(),GaussianNB(),QuadraticDiscriminantAnalysis()]
```

[82]:

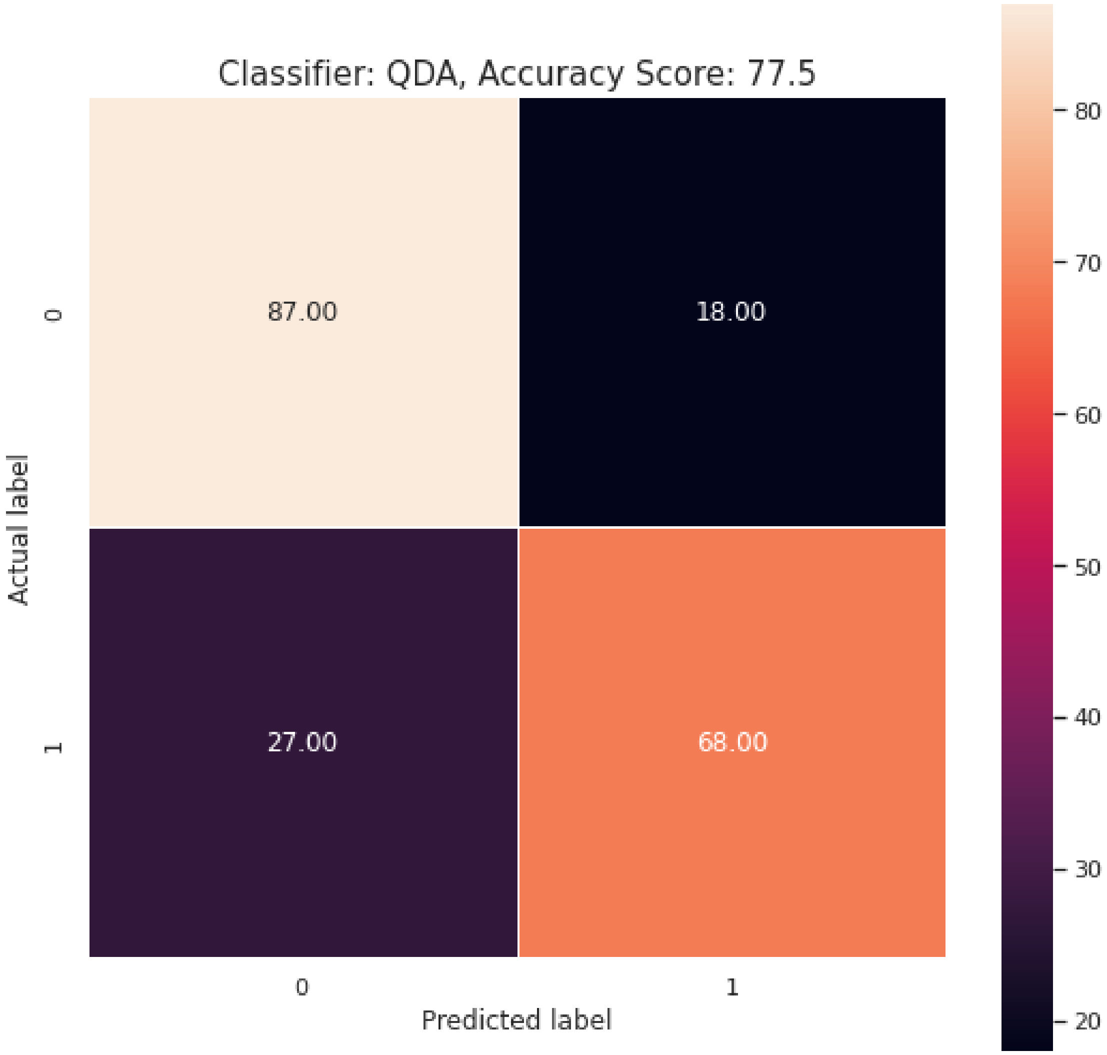
```
# iterate over classifiers
for name, clf in zip(names, classifiers):
#ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
    print("classifier:", name)
clf.fit(X_train, y_train)
score = clf.score(X_test, y_test)
pred = clf.predict(X_test)
prob = clf.predict_proba(X_test)
print(round(score*100, 2))
print(sens_spec(cnf_matrix))
print()
cnf_matrix = confusion_matrix(y_test, pred)
plt.figure(figsize=(9,9))
sns.heatmap(cnf_matrix, annot=True, fmt=".2f", linewidths=.5, square = True);
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
#all_sample_title = 'Accuracy Score: {0}'.format(round(score*100, 2))
all_sample_title ='Classifier: {}, Accuracy Score: {}'.format(name,␣
 ↪round(score*100, 2))
plt.title(all_sample_title, size = 15);
#print()
print(classification_report(y_test, pred))
print()
```

```
skplt.metrics.plot_roc_curve(y_test, prob, figsize = (15, 10), title = 'ROC␣
 ↪Curve for: {}'.format(name))
plt.show()
print('__________________________________________________________________________')
print()
```

```
classifier: Nearest Neighbors
classifier: Logistic Regression
classifier: Linear SVM
classifier: RBF SVM
classifier: Gaussian Process
classifier: Decision Tree
classifier: Random Forest
classifier: Neural Net
classifier: AdaBoost
classifier: Naive Bayes
classifier: QDA
77.5
('Accuracy: 0.78', 'Sensitivity: 0.83', 'Specificity: 0.72')

              precision    recall  f1-score   support

         0.0       0.76      0.83      0.79       105
         1.0       0.79      0.72      0.75        95

    accuracy                           0.78       200
   macro avg       0.78      0.77      0.77       200
weighted avg       0.78      0.78      0.77       200



/usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(*args, **kwargs)
```

```
       ␣
 ↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
 ↪last)

        <ipython-input-86-a16a05fc551c> in <module>
         21 print(classification_report(y_test, pred))
         22 print()
```

```
   ---> 23 sk.metrics.plot_roc_curve(y_test, prob, figsize = (15, 10), title =␣
↪'ROC Curve for: {}'.format(name))
        24 plt.show()
        25␣
↪print('_____________________________________________________________________________________


      /usr/local/lib/python3.7/site-packages/sklearn/utils/validation.py in␣
↪inner_f(*args, **kwargs)
        61             extra_args = len(args) - len(all_args)
        62             if extra_args <= 0:
   ---> 63                 return f(*args, **kwargs)
        64 
        65             # extra_args > 0


      TypeError: plot_roc_curve() missing 1 required positional argument: 'y'
```



[67]:

```
#import random
np.random.seed(1000)
randomlist = []
for i in range(0,10):
    n = np.random.randint(1,len(X_test))
    randomlist.append(n)
    print(randomlist)
```

```
[180]
[180, 88]
[180, 88, 72]
[180, 88, 72, 193]
[180, 88, 72, 193, 95]
[180, 88, 72, 193, 95, 93]
[180, 88, 72, 193, 95, 93, 2]
```

```
[180, 88, 72, 193, 95, 93, 2, 190]
[180, 88, 72, 193, 95, 93, 2, 190, 129]
[180, 88, 72, 193, 95, 93, 2, 190, 129, 90]
```

[68]:

```
list(X_test.iloc[180])
```

[68]:

```
[0.058823529411764705,
 0.3161290322580645,
 0.46938775510204 07,
 0.2608695652173913,
 0.1701499109153931,
 0.24948875255623731,
 0.10119555935098205,
 0.033333333333333326]
```

[69]:

```
y_test.iloc[180]['Outcome']
```

[69]:

```
0.0
```

[70]:

```
pre_out = []
out = []
for i in randomlist:
    data_in = [list(X_test.iloc[i])]
    data_in = np.around(data_in, 2)
    pre_data_out = lr.predict(data_in)
    data_out = y_test.iloc[i]['Outcome']
    mylist = [i, data_in, pre_data_out, data_out]
    print(*mylist,sep='\n')
    print('------------------------')
pre_out.append(pre_data_out)
out.append(data_out)
```

```
180
[[0.06 0.32 0.47 0.26 0.17 0.25 0.1  0.03]]
[0.]
0.0
------------------------
88
[[0.12 0.24 0.49 0.09 0.07 0.24 0.2  0.07]]
[0.]
0.0
------------------------
72
[[0.65 0.49 0.57 0.33 0.16 0.49 0.3  0.45]]
[1.]
1.0
------------------------
193
```

```
[[0.29 0.46 0.51 0.24 0.17 0.15 0.05 0.15]]
[0.]
0.0
------------------------
95
[[0.   0.59 0.45 0.38 0.28 0.49 0.12 0.05]]
[0.]
1.0
------------------------
93
[[0.12 0.41 0.39 0.03 0.32 0.15 0.34 0.02]]
[0.]
0.0
------------------------
2
[[0.82 0.36 0.55 0.2  0.2  0.38 0.14 0.42]]
[0.]
1.0
------------------------
190
[[0.52 0.46 0.57 0.27 0.19 0.32 0.08 0.26]]
[0.]
1.0
------------------------
129
[[0.   0.7  0.59 0.35 0.31 0.48 0.08 0.1 ]]
[1.]
0.0
------------------------
90
[[0.   0.3  0.45 0.27 0.24 0.44 0.13 0.07]]
[0.]
0.0
------------------------

C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
```

```
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
```

[71]:

```
svc = SVC(gamma=2, C=1, probability=True)
svc.fit(X_train, y_train)
```

```
C:\Users\91820\anaconda3\lib\site-packages\sklearn\utils\validation.py:993:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

[71]:

```
SVC(C=1, gamma=2, probability=True)
```

[72]:

```
pre_out = []
out = []
for i in randomlist:
    data_in = [list(X_test.iloc[i])]
    data_in = np.around(data_in, 2)
    pre_data_out = svc.predict(data_in)
    data_out = y_test.iloc[i]['Outcome']
    mylist = [i, data_in, pre_data_out, data_out]
    print(*mylist,sep='\n')
    print('------------------------')
pre_out.append(pre_data_out)
out.append(data_out)
```

```
180
[[0.06 0.32 0.47 0.26 0.17 0.25 0.1  0.03]]
[0.]
0.0
------------------------
88
[[0.12 0.24 0.49 0.09 0.07 0.24 0.2  0.07]]
[0.]
0.0
------------------------
72
[[0.65 0.49 0.57 0.33 0.16 0.49 0.3  0.45]]
[1.]
1.0
------------------------
193
[[0.29 0.46 0.51 0.24 0.17 0.15 0.05 0.15]]
[0.]
0.0
------------------------
95
[[0.   0.59 0.45 0.38 0.28 0.49 0.12 0.05]]
[1.]
1.0
------------------------
93
[[0.12 0.41 0.39 0.03 0.32 0.15 0.34 0.02]]
[0.]
0.0
------------------------
2
[[0.82 0.36 0.55 0.2  0.2  0.38 0.14 0.42]]
[1.]
1.0
------------------------
190
[[0.52 0.46 0.57 0.27 0.19 0.32 0.08 0.26]]
[1.]
1.0
------------------------
129
[[0.   0.7  0.59 0.35 0.31 0.48 0.08 0.1 ]]
[1.]
```

```
0.0
------------------------
90
[[0.   0.3  0.45 0.27 0.24 0.44 0.13 0.07]]
[0.]
0.0
------------------------

C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
C:\Users\91820\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

1 Our Basic Logistic Regression algorithm is

predicting 6 / 10 inputs correctly, whereas the best performing SVC algorithm is predicting 9 / 10 inputs correctly

```
[ ]:
```

```
[ ]:
```

[ ]:

```
```