In [1]:

```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

# Visualization Library
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

# Robust Scaler
from sklearn.preprocessing import RobustScaler

# Synthetic Minority Over-sampling Technique
from imblearn.over_sampling import SMOTE

# Hold-out Method
from sklearn.model_selection import train_test_split

# Grid Search CV
from sklearn.model_selection import GridSearchCV

# Logistic Regression
from sklearn.linear_model import LogisticRegression

# Tree Model
from sklearn.tree import DecisionTreeClassifier

# Support Vector Machine
from sklearn.svm import SVC

# Ensemble Model
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier

# Metrics
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.metrics import accuracy_score
```

In [2]:

```
dataset = pd.read_csv('health care diabetes.csv')
df = dataset.copy() #Creating the copy of the dataframe
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0

4	0	137	40	35	168	43.1	2.288	33	1

In [3]:

```
df.shape
```

Out[3]:

```
(768, 9)
```

Descriptive Analysis

In [4]:

```
df.describe()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.0
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.3
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.4
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.0
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.0
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.0
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.0
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.0

In [5]:

```
class_0 = df[df.Outcome ==0] # Outcome = 0 (i.e) Non-Diabetic Patient
class_1 = df[df.Outcome ==1] # Outcome = 1 (i.e) Diabetic Patient
```

In [6]:

```
outcome_0 = round(class_0.describe(),2)
outcome_1 = round(class_1.describe(),2)
```

In [7]:

```
outcome_0
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.0
mean	3.30	109.98	68.18	19.66	68.79	30.30	0.43	31.19	0.0
std	3.02	26.14	18.06	14.89	98.87	7.69	0.30	11.67	0.0
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	21.00	0.0
25%	1.00	93.00	62.00	0.00	0.00	25.40	0.23	23.00	0.0
50%	2.00	107.00	70.00	21.00	39.00	30.05	0.34	27.00	0.0
75%	5.00	125.00	78.00	31.00	105.00	35.30	0.56	37.00	0.0

max	13.00	197.00	122.00	60.00	744.00	57.30	2.33	81.00	0.0

Outcome_0 Descriptive Analysis Insights

For **Pregnancies**, Mean is 3.3 and upto 75 percentile it shows 5 and Max it shows 13, it shows that the persence of outlier.

For **Insulin**, Mean is 68.79, Min and 25 percentile it is 0, @ 75 percentile it is 105, but in Max it shows 744, it shows the presence of outliers.

In [8]:

```
outcome_1
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	268.00	268.00	268.00	268.00	268.00	268.00	268.00	268.00	268.0
mean	4.87	141.26	70.82	22.16	100.34	35.14	0.55	37.07	1.0
std	3.74	31.94	21.49	17.68	138.69	7.26	0.37	10.97	0.0
min	0.00	0.00	0.00	0.00	0.00	0.00	0.09	21.00	1.0
25%	1.75	119.00	66.00	0.00	0.00	30.80	0.26	28.00	1.0
50%	4.00	140.00	74.00	27.00	0.00	34.25	0.45	36.00	1.0
75%	8.00	167.00	82.00	36.00	167.25	38.78	0.73	44.00	1.0
max	17.00	199.00	114.00	99.00	846.00	67.10	2.42	70.00	1.0

Outcome_1 Descriptive Analysis Insights

For **Pregnancies**, Mean is 4.87, Median or 50 percentile is 4 counts, but Max counts 17

For **Insulin**, Mean is 100.84, 25 percentile and median/50 percentile is 0, But Max it shows 846. It shows the persence of Outliers

Univariate Analysis

In [9]:

```
class_0 = class_0.drop('Outcome', axis=1)
```

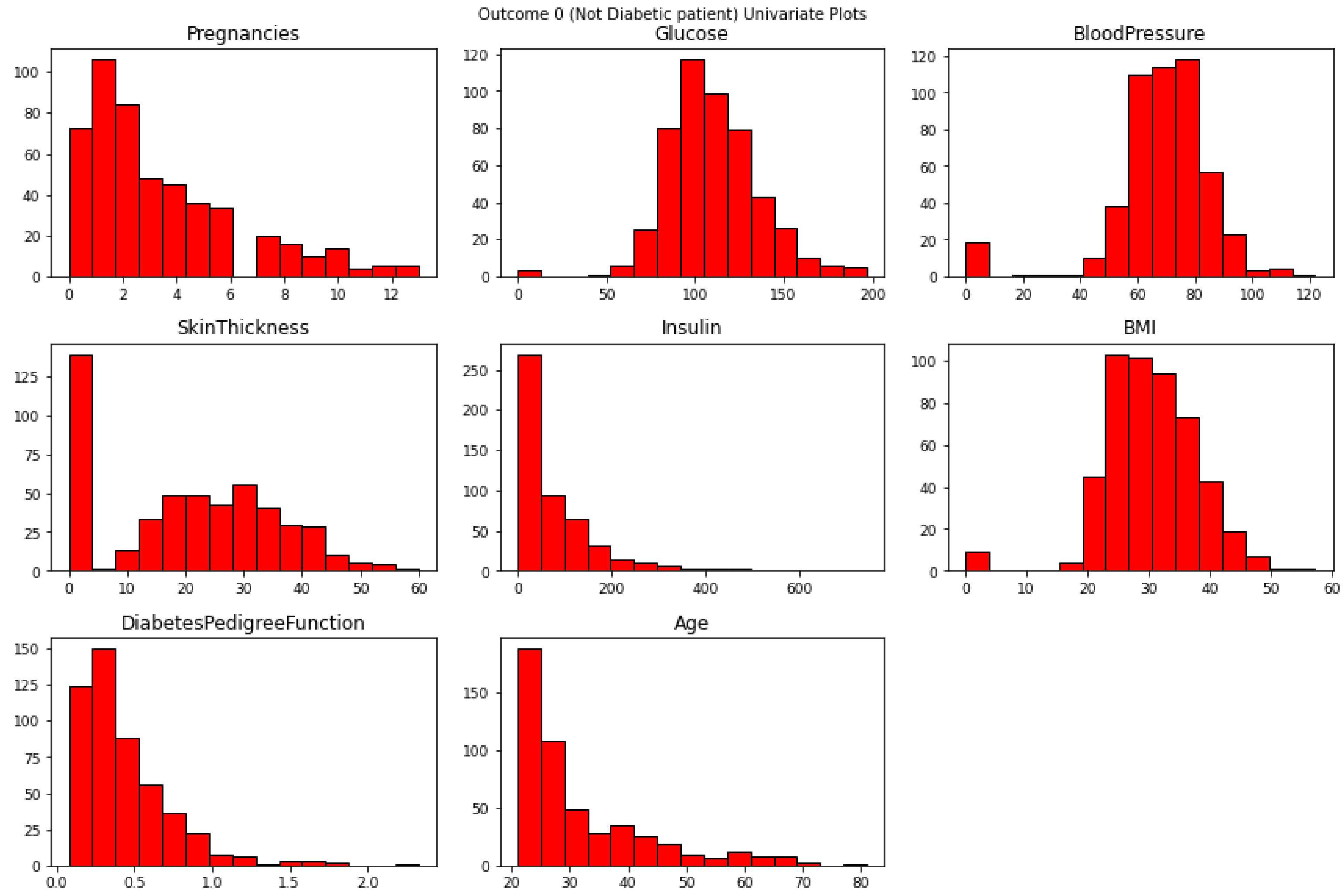
In [12]:

```
class_0.hist(bins=15,
             color='red',
             edgecolor='black',
             linewidth=1.0,
             xlabelsize=9,
             ylabelsize=9,
             grid=False)

plt.tight_layout(rect=(0, 0, 2, 2)) # it will change the size of the plot

plt.suptitle('Outcome 0 (Not Diabetic patient) Univariate Plots',
             x=1, # title x position
```

```
                y=2, # title y position
                fontsize=10)
```

Out[12]:

```
Text(1, 2, 'Outcome 0 (Not Diabetic patient) Univariate Plots')
```



Here BMI, Blood Pressure, Glucose, Skin Thickness are almost Normally or Gaussian Distribution.

Here the Age, Insulin, Pregnancies, DiabetesPedigreeFunction the data is **right skewed**.

Some datas are loaded majorly on 0.

In [13]:

```
class_1 = class_1.drop('Outcome',axis=1)
```

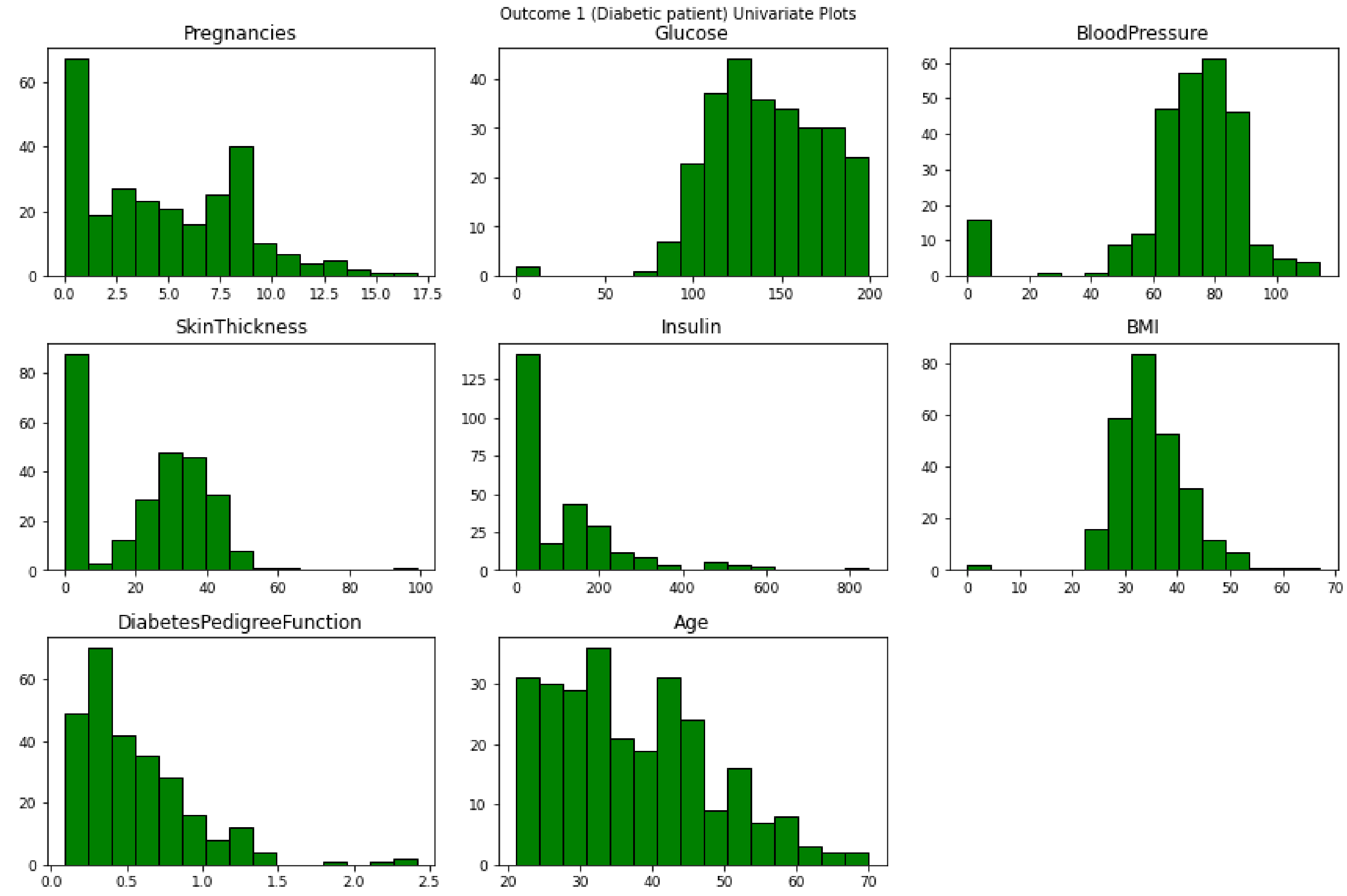
In [14]:

```
class_1.hist(bins=15,
             color='green',
             edgecolor='black',
             linewidth=1.0,
             xlabelsize=9,
             ylabelsize=9,
             grid=False)

plt.tight_layout(rect=(0, 0, 2, 2)) # it will change the size of the plot

plt.suptitle('Outcome 1 (Diabetic patient) Univariate Plots',
             x=1, # x position of title
             y=2, # y position of title
```

```
                fontsize=10)
```

Out[14]:

```
Text(1, 2, 'Outcome 1 (Diabetic patient) Univariate Plots')
```



Here BMI, Blood Pressure, Skin Thickness are almost Normally or Gaussian Distribution.

Here the Age, Insulin, Pregnancies, DiabetesPedigreeFunction the data is **right skewed**.

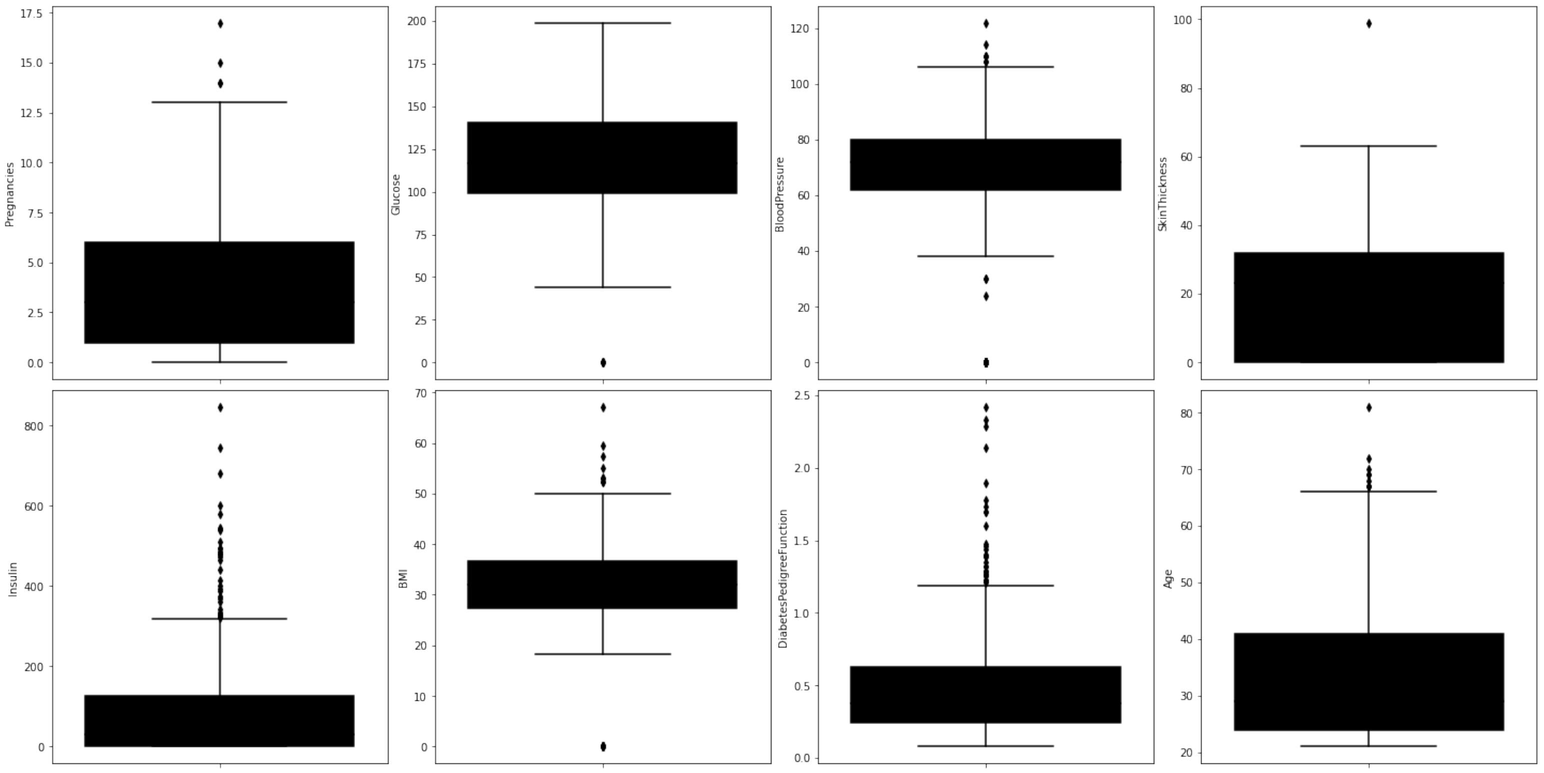
Some datas are loaded majorly on 0.

In [15]:

```
df_box = df.drop('Outcome',axis=1)
```

In [16]:

```
fig , ax = plt.subplots(nrows= 2,       # no,of plots comes in row wise
                        ncols= 4,       # no,of plots comes in column wise
                        figsize=(20,10) # size of plot
                        )
ax = ax.flatten() # It returns a flattened version of the array, to avoid numpy.ndarray
index = 0
for i in df_box.columns:
  sns.boxplot(y=i,data = df_box, ax=ax[index],color='black')
  index += 1
plt.tight_layout(pad=0.4)
```



In Box plot, it completely shows the picture of the Outlier present in the datasets.

All the series of data column has outlier, since the shape of the data (768, 9), it shows that datasets is very small and the outliers cannot be removed.

But it should be scaled using Robust scaler because it consists of many outlier where Standard Scaler, Min max Scaler etc are sensitive to outliers

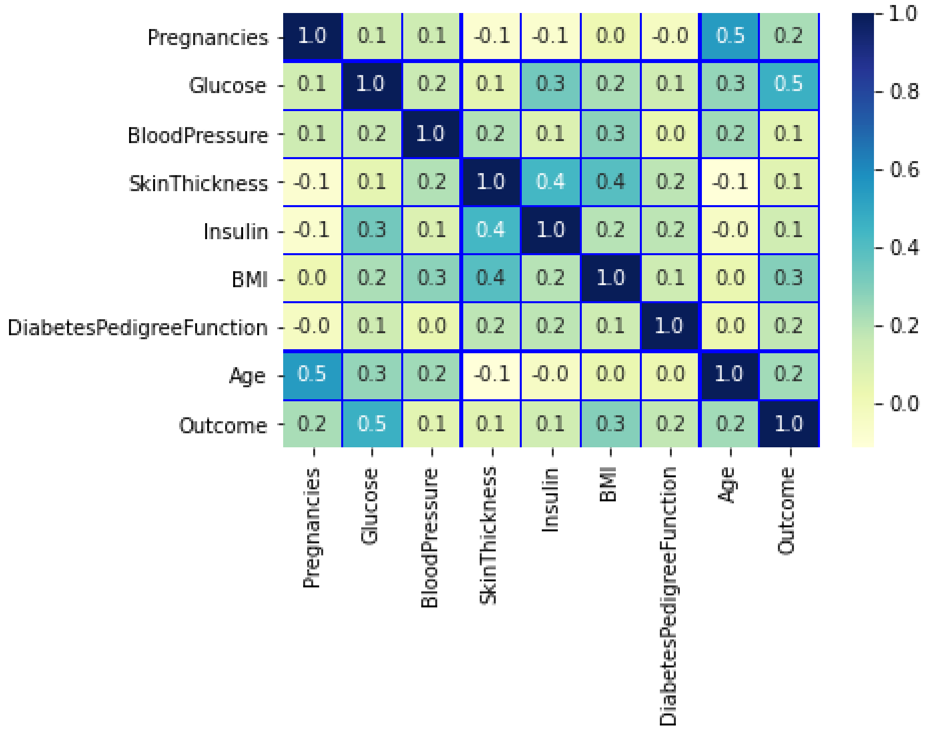
Multivariate Analysis

In [17]:

```
corr = df.corr()
sns.heatmap(corr,
            fmt='.1f',
            linewidth=0.2,
            linecolor='blue',
            annot = True,
            cmap="YlGnBu"
            )
```

Out[17]:

```
<AxesSubplot:>
```



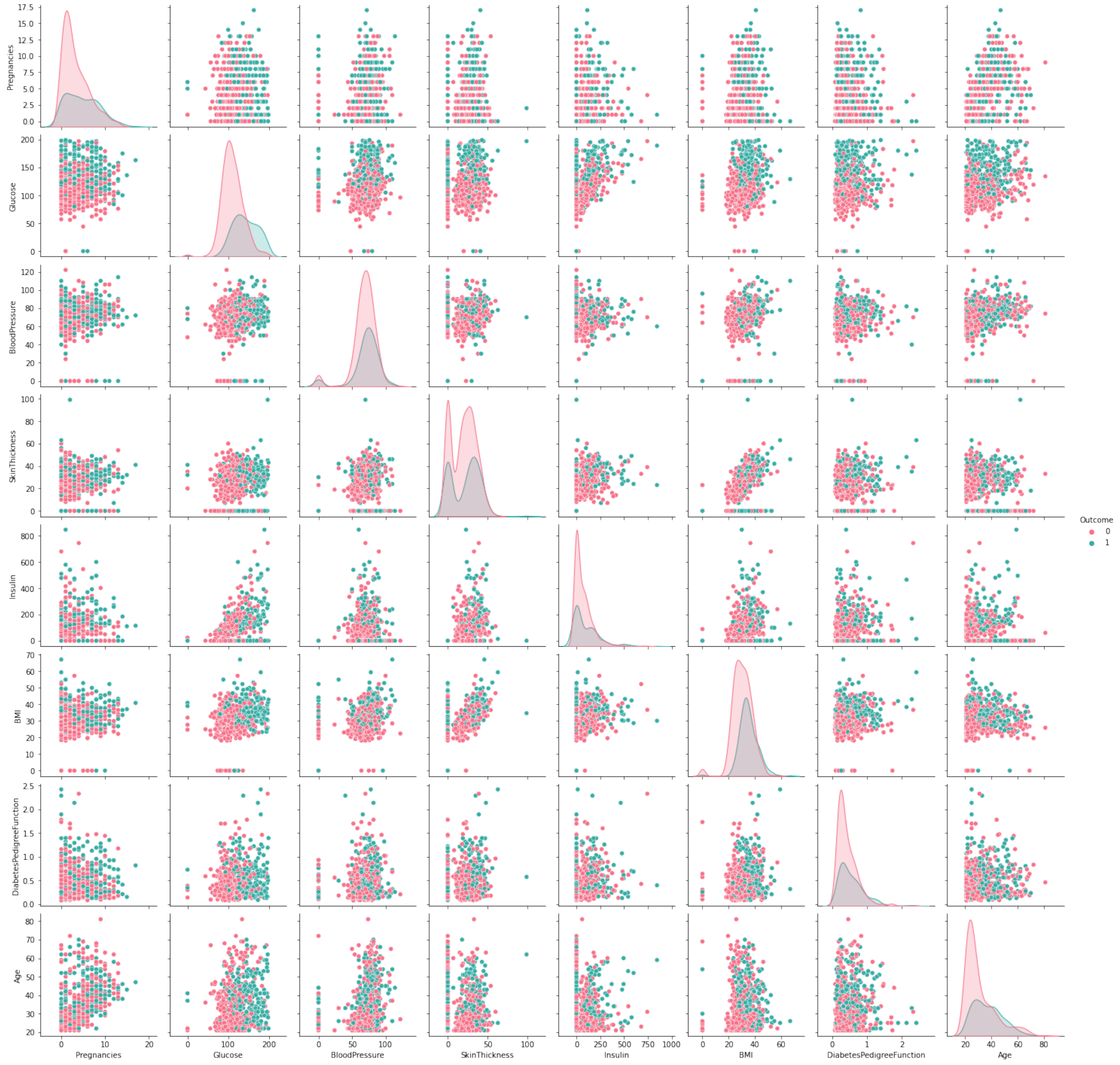
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.0	0.1	0.1	-0.1	-0.1	0.0	-0.0	0.5	0.2
Glucose	0.1	1.0	0.2	0.1	0.3	0.2	0.1	0.3	0.5
BloodPressure	0.1	0.2	1.0	0.2	0.1	0.3	0.0	0.2	0.1
SkinThickness	-0.1	0.1	0.2	1.0	0.4	0.4	0.2	-0.1	0.1
Insulin	-0.1	0.3	0.1	0.4	1.0	0.2	0.2	-0.0	0.1
BMI	0.0	0.2	0.3	0.4	0.2	1.0	0.1	0.0	0.3
DiabetesPedigreeFunction	-0.0	0.1	0.0	0.2	0.2	0.1	1.0	0.0	0.2
Age	0.5	0.3	0.2	-0.1	-0.0	0.0	0.0	1.0	0.2
Outcome	0.2	0.5	0.1	0.1	0.1	0.3	0.2	0.2	1.0

In [18]:

```
sns.pairplot(df,                # dataset
             hue='Outcome',     # variable in dataset to map plot aspects to different color
             palette='husl',
             )
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x1d26bc2ed60>
```



Above plot shows the distribution of dataset in scatter plot and Kernel Density Estimator, for the different combination of datasets with respect to the Outcome.

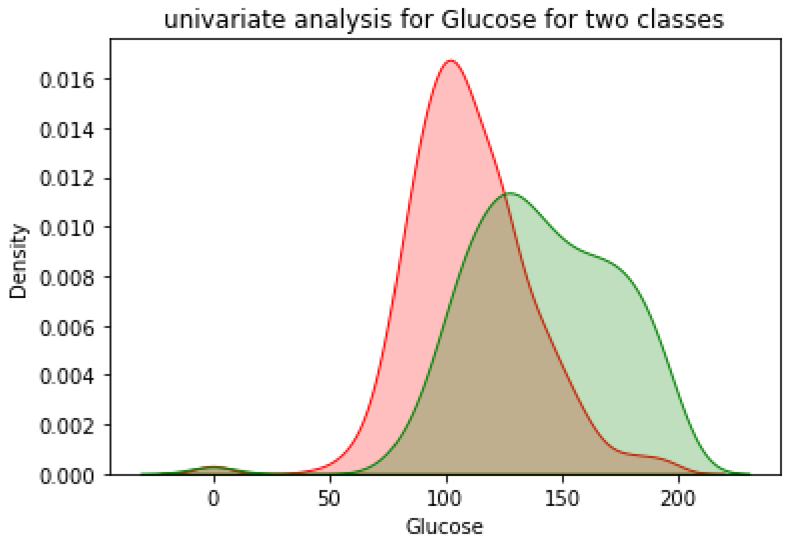
Bivariate Analysis

In [19]:

```
sns.kdeplot(class_0.Glucose, shade=True,color="r")
sns.kdeplot(class_1.Glucose, shade=True,color="g")
plt.title('univariate analysis for Glucose for two classes')
```

Out[19]:

```
Text(0.5, 1.0, 'univariate analysis for Glucose for two classes')
```



Here In this Kernel Density Estimate plot, the data of Glucose for Outcome_0 is distributed like a Normal/Gaussian distribution, with sharp peakness and most of the data points accumalted nearby the mean.

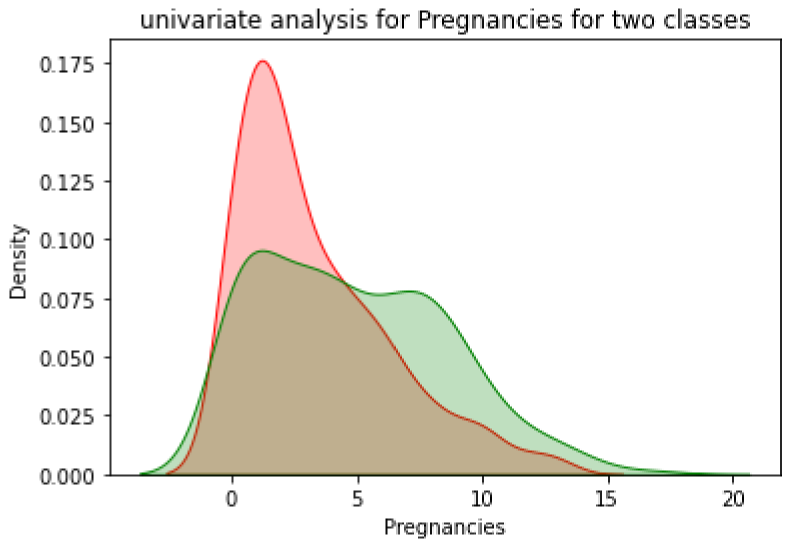
But the curve looks abnormal because of the Outcome_1, here the datas are accumaleted in a wide pattern with some outliers at the bottom.

In [22]:

```
sns.kdeplot(class_0.Pregnancies, shade=True,color="r")
sns.kdeplot(class_1.Pregnancies, shade=True,color="g")
plt.title('univariate analysis for Pregnancies for two classes')
```

Out[22]:

```
Text(0.5, 1.0, 'univariate analysis for Pregnancies for two classes')
```



Here in the above 2 curves, In Outcome_0 (red curve), the Pregnancies distribution is look like right skewed, with sharp peakness and mode is one time Pregnancy counts around 106, the median is two times pregnancy with the count of 84.

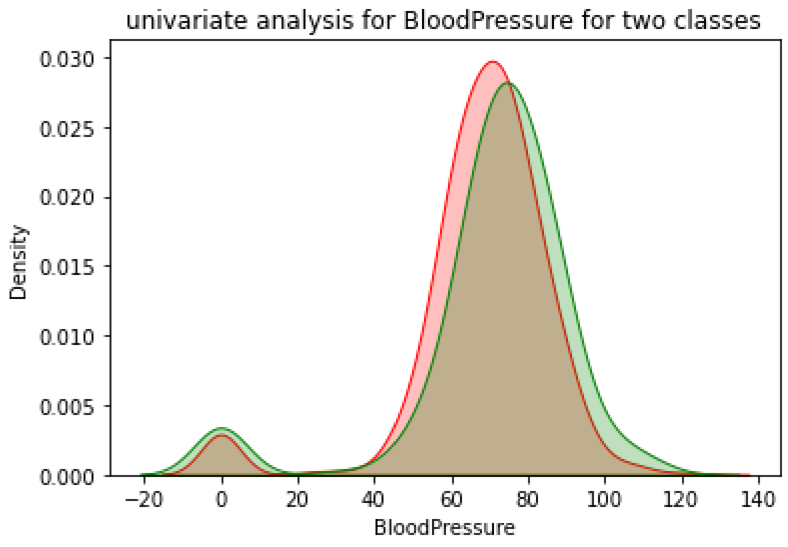
In Outcome_1 (Green curve), the Pregnancies distribution is look like abnormal distribution, data is distributed widely and mode is zero time Pregnancy counts around 38, the median is four times pregnancy with the count of 23. The right tail extended widely at end.

In [23]:

```
sns.kdeplot(class_0.BloodPressure, shade=True,color="r")
sns.kdeplot(class_1.BloodPressure, shade=True,color="g")
plt.title('univariate analysis for BloodPressure for two classes')
```

Out[23]:

```
Text(0.5, 1.0, 'univariate analysis for BloodPressure for two classes')
```



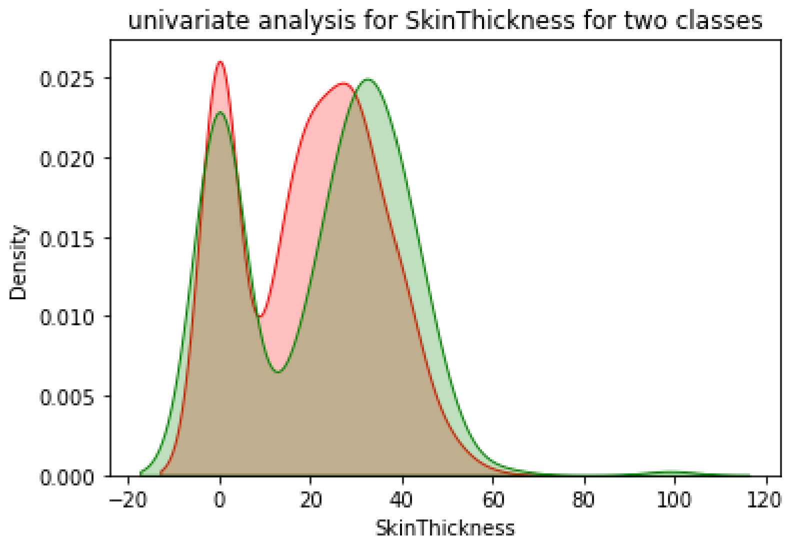
Here in above kDE plot, for the both the Outcome 1 & 0, except that small abnormality near zero, the rest of the data is completely shows that it is normal distributed

In [24]:

```
sns.kdeplot(class_0.SkinThickness,shade=True,color="r")
sns.kdeplot(class_1.SkinThickness, shade=True,color="g")
plt.title('univariate analysis for SkinThickness for two classes')
```

Out[24]:

```
Text(0.5, 1.0, 'univariate analysis for SkinThickness for two classes')
```



Here in both the curve the mode is equal to 0, for Outcome = 0 it counts upto 139 and Outcome = 1 it counts upto 88, this make the curve looks so abnormal.

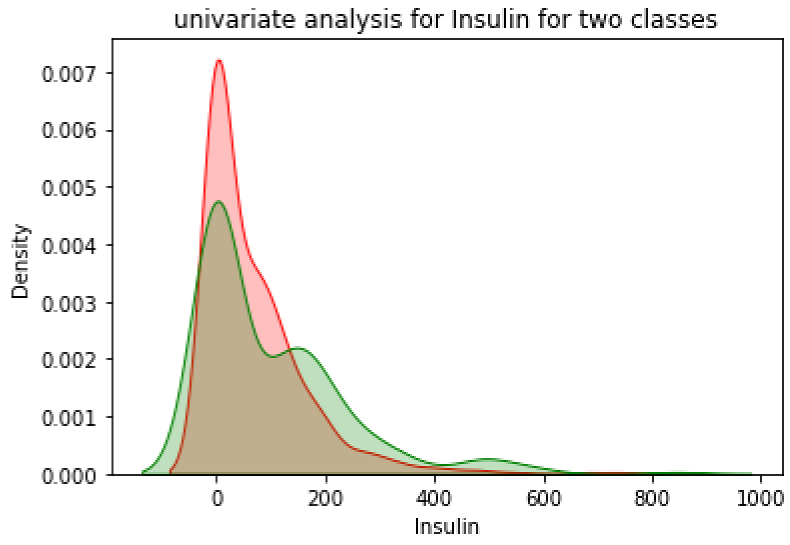
The tail part of Outcome = 0, is extended quite alot. The presence of outlier will distrub the classfication while building the model i.e it will result in poor classification or overfitting.

In [25]:

```
sns.kdeplot(class_0.Insulin,shade=True,color="r")
sns.kdeplot(class_1.Insulin, shade=True,color="g")
plt.title('univariate analysis for Insulin for two classes')
```

Out[25]:

```
Text(0.5, 1.0, 'univariate analysis for Insulin for two classes')
```



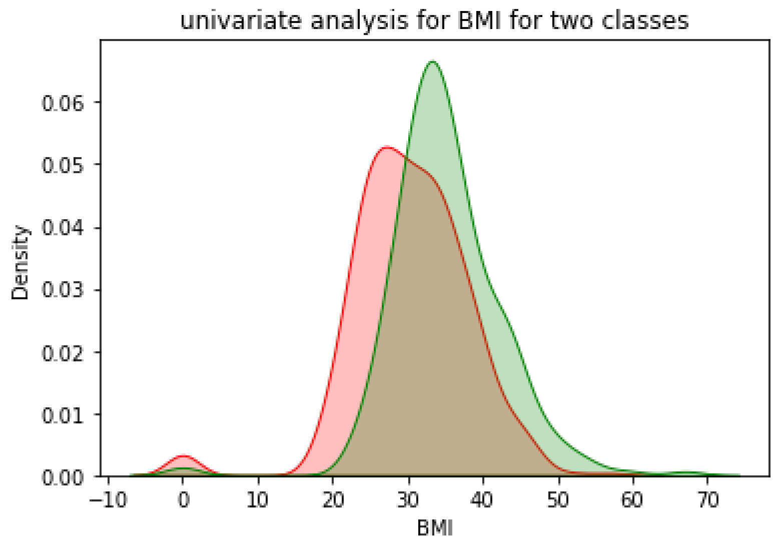
KDE plot for both the Outcome is completely right skewed. The data distribtion of Outcome_1 is like so wavy and the right tail is extended & shows the plots of outliers.

In [26]:

```
sns.kdeplot(class_0.BMI,shade=True,color="r")
sns.kdeplot(class_1.BMI, shade=True,color="g")
plt.title('univariate analysis for BMI for two classes')
```

Out[26]:

```
Text(0.5, 1.0, 'univariate analysis for BMI for two classes')
```



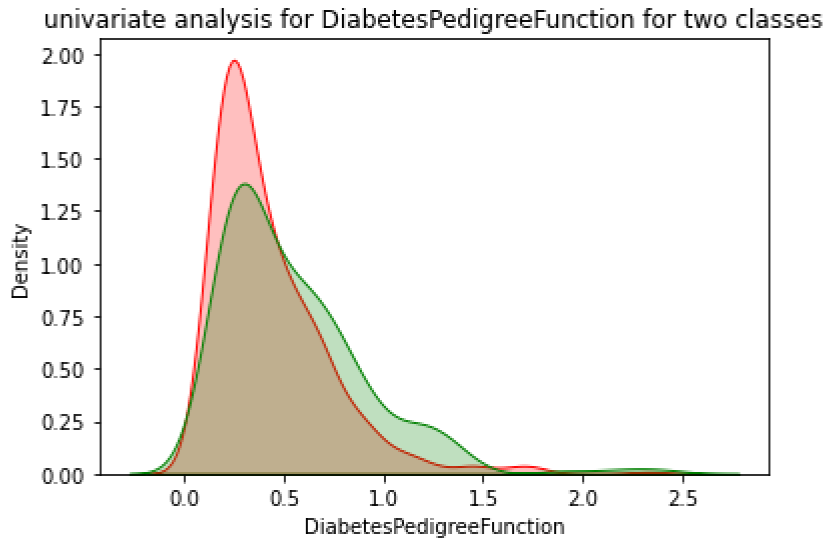
Here, there is some accumulation of data on left side of both the curve. But the Outcome_1 is a normal distribution and it tell BMI plays one important role in Diabetes, abnormal is BMI will causes Diabetes and the right tail keep extending.

In [27]:

```
sns.kdeplot(class_0.DiabetesPedigreeFunction,shade=True,color="r")
sns.kdeplot(class_1.DiabetesPedigreeFunction, shade=True,color="g")
plt.title('univariate analysis for DiabetesPedigreeFunction for two classes')
```

Out[27]:

```
Text(0.5, 1.0, 'univariate analysis for DiabetesPedigreeFunction for two classes')
```



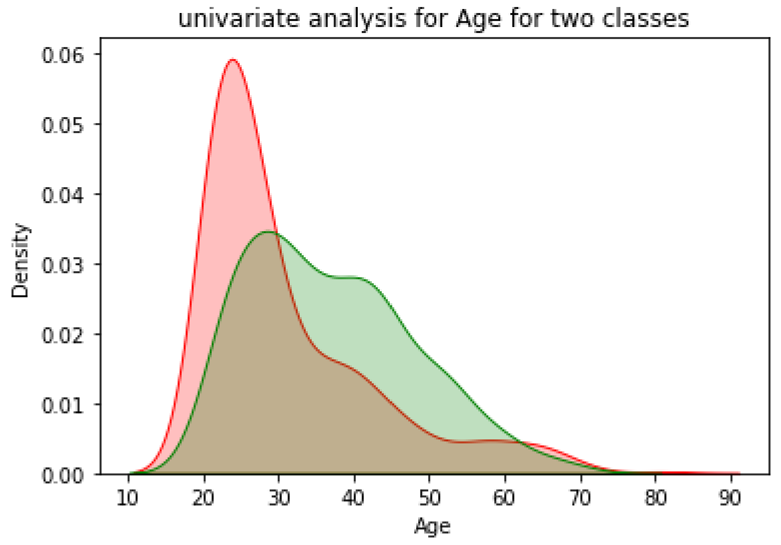
Here in above curve, it shows the picture of both the outcomes were right skewed. Outcome 1, the data distribution is wide so there is abnormal in shape of Density for Outcome_1.

In [28]:

```
sns.kdeplot(class_0.Age,shade=True,color="r")
sns.kdeplot(class_1.Age, shade=True,color="g")
plt.title('univariate analysis for Age for two classes')
```

Out[28]:

```
Text(0.5, 1.0, 'univariate analysis for Age for two classes')
```



KDE plot for Age, outcome_0 is right skewed of mode value of 22 with the counts of 61 and median of 27 with the counts of 24. But after the age of 35 (approx) the distribution pattern is abnormal towards the right tail.

KDE plot for Age, outcome_1 the data is speard across between 25-50 yrs.

In [29]:

```
'''
sns.pairplot(df,                # dataset
             hue='Outcome',     # variable in dataset to map plot aspects to different
color
             kind = 'reg'       # kind of chart
             )'''
```

Out[29]:

```
"\nsns.pairplot(df,               # dataset\n             hue='Outcome',    # variable in da
taset to map plot aspects to different color\n               kind = 'reg'      # kind of chart
\n              )"
```

Here in above pair plot gives the insights of how the data distribution and relationship between the two variable in scatter plot for both the classes, how it the shows the liner regressionline for both the classes.

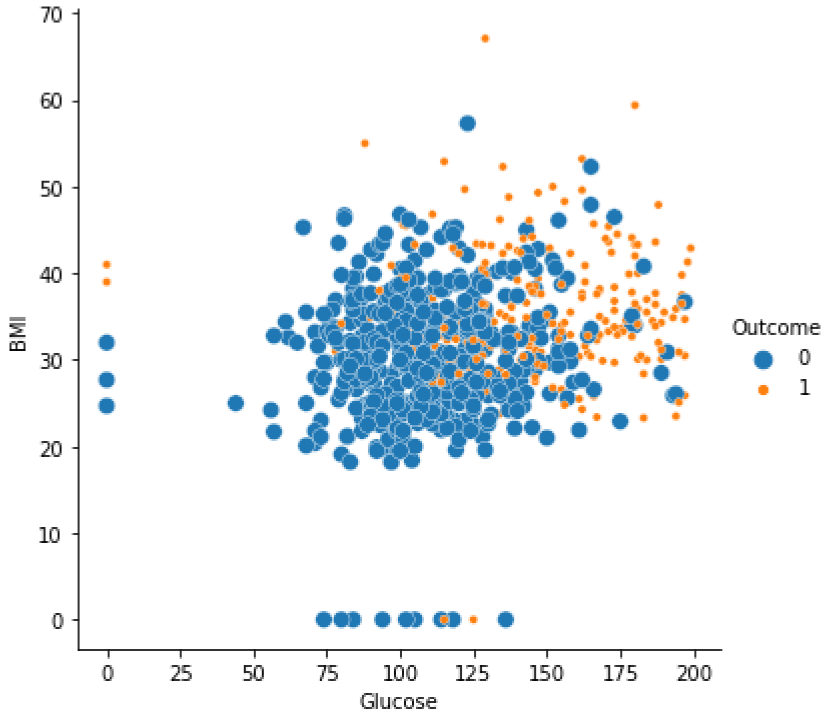
It tell that the data needs to be scaled properly, by considering its outlier as well.

In [30]:

```
sns.relplot(x='Glucose',
            y='BMI',
            data = df,
            hue = 'Outcome',
            size='Outcome')
```

Out[30]:

```
<seaborn.axisgrid.FacetGrid at 0x1d26eac7dc0>
```



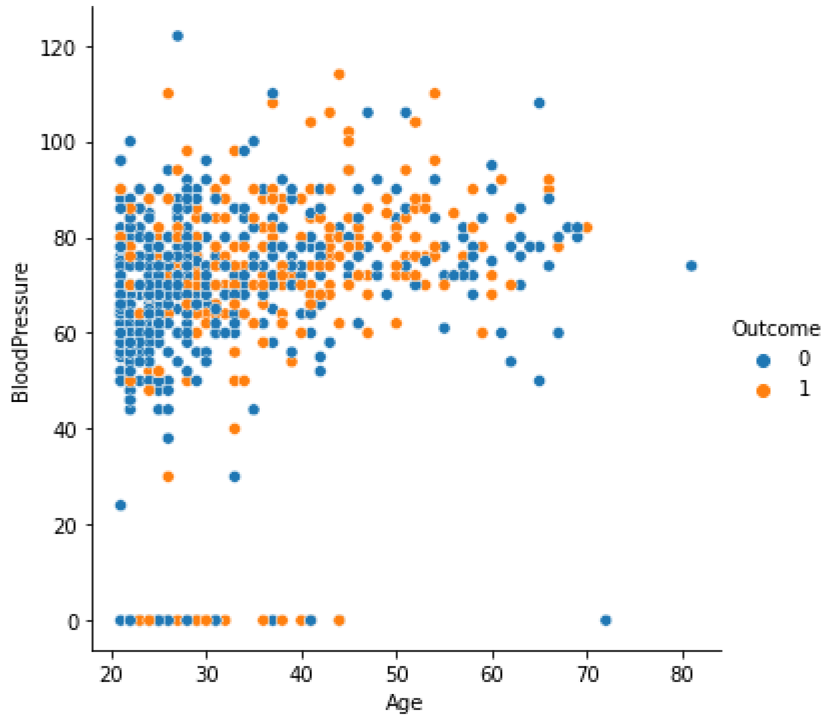
The above scatter plot tells, the people who have abnormal BMI and higher the Glucose level, will have higher the chance of getting Diabetic(orange small dots)

In [31]:

```
sns.relplot(x='Age',
            y='BloodPressure',
            data = df,
            hue = 'Outcome',)
            #size='Outcome')
```

Out[31]:

```
<seaborn.axisgrid.FacetGrid at 0x1d26eddecd0>
```



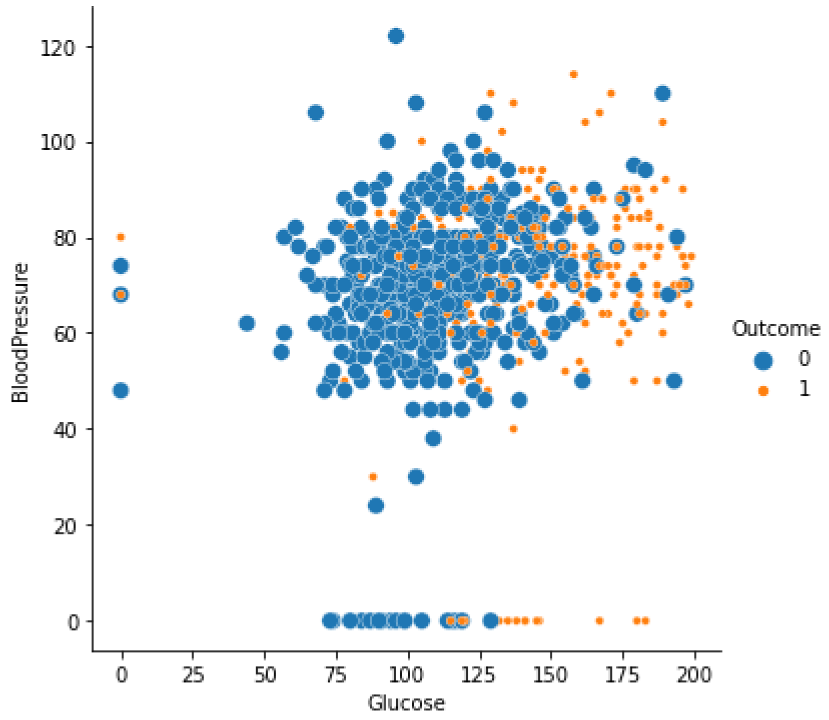
Here the above plot shows that, higher the chance for people have High BloodPressure and getting Aged to be a Diabetic.

In [32]:

```
sns.relplot(x='Glucose',
            y='BloodPressure',
            data = df,
            hue = 'Outcome',
            size='Outcome')
```

Out[32]:

```
<seaborn.axisgrid.FacetGrid at 0x1d26eceaa90>
```



Here the above plot shows that, higher the chance for people have High BloodPressure and Glucose to be a Diabetic.

Feature Tranformation

Binning the age to avoid the model to get distrubed by outliers.

In [33]:

```
df['Age_bin']=pd.cut(x = df['Age'],                                     # Cloumn to be
binned
                        bins = [20,30,50,100],                          # Binnning sizes
                        labels = ['young_aged','middle_aged','old_aged'] # class name for
Binning
                   )
df.head(9)
```

Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Age_bin
0	6	148	72	35	0	33.6	0.627	50	1	middle_aged
1	1	85	66	29	0	26.6	0.351	31	0	middle_aged
2	8	183	64	0	0	23.3	0.672	32	1	middle_aged
3	1	89	66	23	94	28.1	0.167	21	0	young_aged
4	0	137	40	35	168	43.1	2.288	33	1	middle_aged
5	5	116	74	0	0	25.6	0.201	30	0	young_aged
6	3	78	50	32	88	31.0	0.248	26	1	young_aged
7	10	115	0	0	0	35.3	0.134	29	0	young_aged
8	2	197	70	45	543	30.5	0.158	53	1	old_aged

In [34]:

```
df = df.drop('Age',axis=1)
df.head()
```

Out[34]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	Age_bin
0	6	148	72	35	0	33.6	0.627	1	middle_aged
1	1	85	66	29	0	26.6	0.351	0	middle_aged
2	8	183	64	0	0	23.3	0.672	1	middle_aged
3	1	89	66	23	94	28.1	0.167	0	young_aged
4	0	137	40	35	168	43.1	2.288	1	middle_aged

In [35]:

```
Age_bin = df.Age_bin
df.insert(0,'Aged_bin',Age_bin)
df = df.drop('Age_bin',axis=1)
df.head()
```

Out[35]:

	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome
0	middle_aged	6	148	72	35	0	33.6	0.627	1

1	middle_aged	1	85	66	29	0	26.6	0.351	0
2	middle_aged	8	183	64	0	0	23.3	0.672	1
3	young_aged	1	89	66	23	94	28.1	0.167	0
4	middle_aged	0	137	40	35	168	43.1	2.288	1

Encoding the categorical Age column

In [36]:

```
df.Aged_bin = df.Aged_bin.replace(to_replace =
['young_aged','middle_aged','old_aged'],value =[0,1,2],inplace=False)
df.head()
```

Out[36]:

	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome
0	1	6	148	72	35	0	33.6	0.627	1
1	1	1	85	66	29	0	26.6	0.351	0
2	1	8	183	64	0	0	23.3	0.672	1
3	0	1	89	66	23	94	28.1	0.167	0
4	1	0	137	40	35	168	43.1	2.288	1

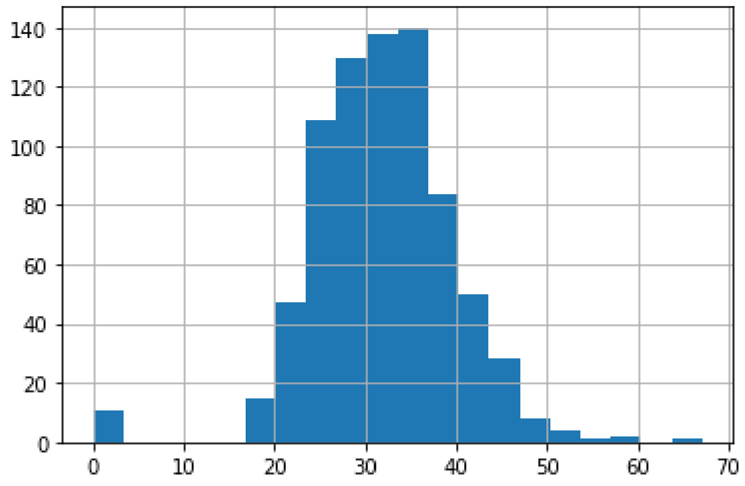
Treating BMI Column

In [37]:

```
df.BMI.hist(bins=20)
```

Out[37]:

```
<AxesSubplot:>
```



In [38]:

```
df.loc[((df.Outcome == 0) &  (df.BMI ==0)) , 'BMI'] = df.loc[((df.Outcome == 0) &  (df.BMI
==0)) , 'BMI'].mean()
```

In [39]:

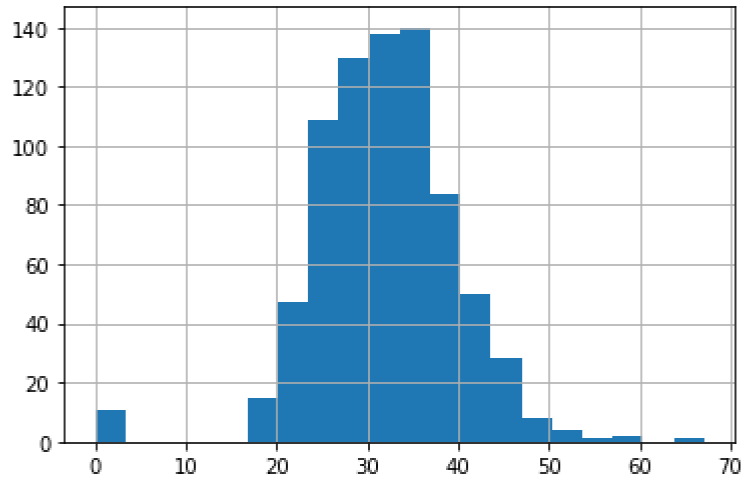
```
df.loc[((df.Outcome == 1) &  (df.BMI ==0)) , 'BMI'] = df.loc[((df.Outcome == 1) &  (df.BMI
==0)) , 'BMI'].mean()
```

In [40]:

```
df.BMI.hist(bins=20)
```

Out[40]:

```
<AxesSubplot:>
```



In [41]:

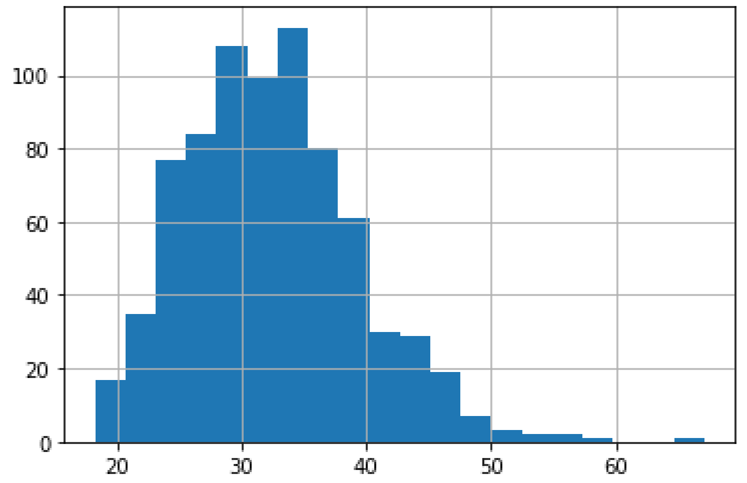
```
class_0 = (df['Outcome'] == 0)
class_1 = (df['Outcome'] == 1)
avg_class_0 = df.loc[class_0, 'BMI'].mean()
avg_class_1 = df.loc[class_1, 'BMI'].mean()
df.loc[df['BMI']==0 & class_0, 'BMI'] = avg_class_0
df.loc[df['BMI']==0 & class_1, 'BMI'] = avg_class_1
```

In [42]:

```
df.BMI.hist(bins=20)
```

Out[42]:

```
<AxesSubplot:>
```

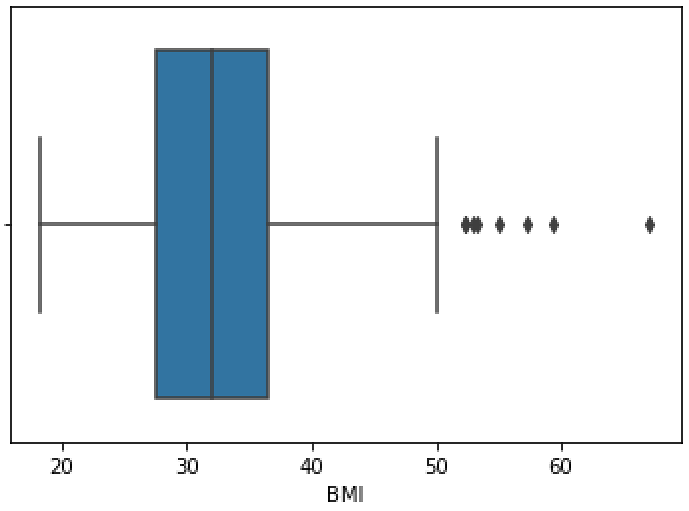


In [43]:

```
sns.boxplot(x=df.BMI)
```

Out[43]:

```
<AxesSubplot:xlabel='BMI'>
```



Category BMI range - kg/m2 Severe Thinness < 16 Moderate Thinness 16 - 17 Mild Thinness 17 - 18.5 Normal 18.5 - 25 Overweight 25 - 30 Obese Class I 30 - 35 Obese Class II 35 - 40 Obese Class III > 40

Since it has many outlier, So it is converted into a Categorical caloumn

Binning the BMI Column

In [44]:

```
df['BMI_bin']=pd.cut(x = df['BMI'],
                      bins = [18,25,30,80],
                      labels = ['Normal','Overweight','Obese'])
df.head(9)
```

Out[44]:

	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	BMI_bin
0	1	6	148	72	35	0	33.6	0.627	1	Obese
1	1	1	85	66	29	0	26.6	0.351	0	Overweight
2	1	8	183	64	0	0	23.3	0.672	1	Normal
3	0	1	89	66	23	94	28.1	0.167	0	Overweight
4	1	0	137	40	35	168	43.1	2.288	1	Obese
5	0	5	116	74	0	0	25.6	0.201	0	Overweight
6	0	3	78	50	32	88	31.0	0.248	1	Obese
7	0	10	115	0	0	0	35.3	0.134	0	Obese
8	2	2	197	70	45	543	30.5	0.158	1	Obese

In [45]:

```
df.BMI_bin = df.BMI_bin.replace(to_replace = ['Normal','Overweight','Obese'],value
=[0,1,2],inplace=False)
df.head()
```

Out[45]:

	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Outcome	BMI_bin
0	1	6	148	72	35	0	33.6	0.627	1	2
1	1	1	85	66	29	0	26.6	0.351	0	1
2	1	8	183	64	0	0	23.3	0.672	1	0

3	0	1	89	66	23	94	28.1	0.167	0	1
4	1	0	137	40	35	168	43.1	2.288	1	2

In [46]:

```
BMI_bin = df.BMI_bin
df.insert(0,'BMI_binned',BMI_bin)
```

In [47]:

```
df = df.drop(['BMI_bin','BMI'],axis=1)
df.head()
```

Out[47]:

	BMI_binned	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	DiabetesPedigreeFunction	Outcome
0	2	1	6	148	72	35	0	0.627	1
1	1	1	1	85	66	29	0	0.351	0
2	0	1	8	183	64	0	0	0.672	1
3	1	0	1	89	66	23	94	0.167	0
4	2	1	0	137	40	35	168	2.288	1

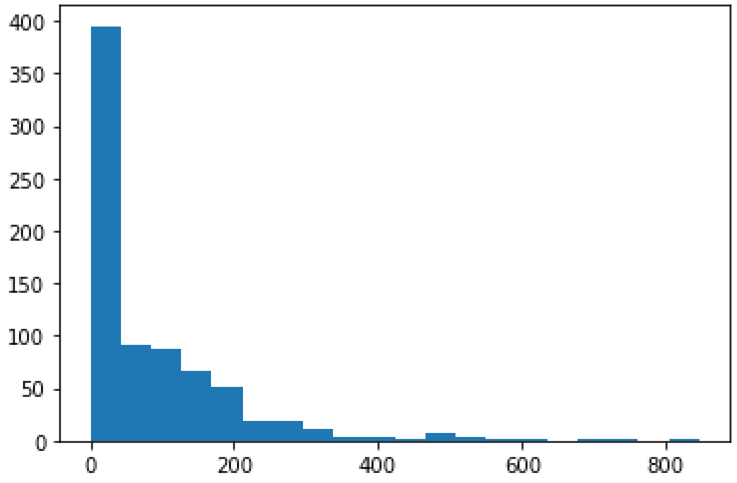
Treating the Insulin

In [48]:

```
df.Insulin.hist(bins=20,grid=False)
```

Out[48]:

```
<AxesSubplot:>
```



In [49]:

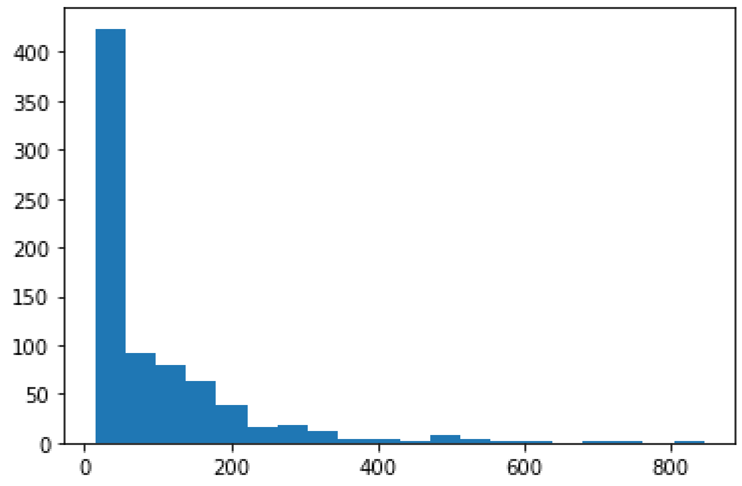
```
class_0 = (df['Outcome'] == 0)
class_1 = (df['Outcome'] == 1)
avg_class_0 = df.loc[class_0,'Insulin'].median()
avg_class_1 = df.loc[class_1,'Insulin'].median()
df.loc[df['Insulin']==0 & class_0, 'Insulin'] = avg_class_0
df.loc[df['Insulin']==0 & class_1, 'Insulin'] = avg_class_1
```

In [50]:

```
df.Insulin.hist(bins=20,grid=False)
```

Out[50]:

```
<AxesSubplot:>
```



Here in above plot, even after replacing the 0 with median values based on their respective class, still it looks right skewed.

Log() Tranformation

In [51]:

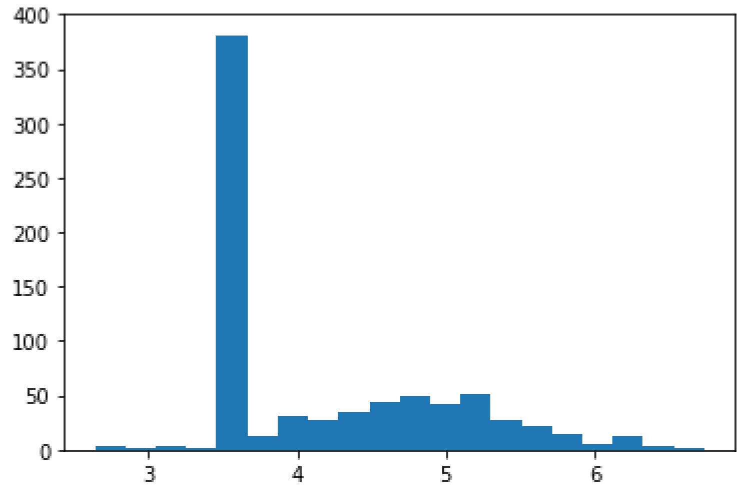
```
df.Insulin = np.log(df.Insulin)
```

In [52]:

```
df.Insulin.hist(bins=20,grid=False)
```

Out[52]:

```
<AxesSubplot:>
```

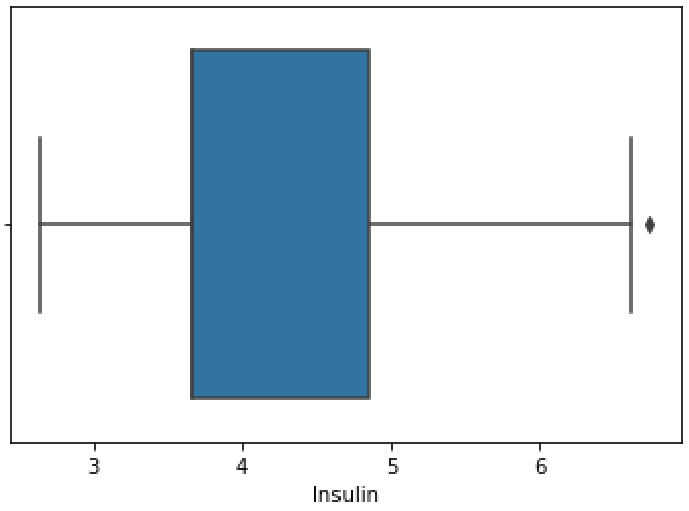


In [53]:

```
sns.boxplot(x=df.Insulin)
```

Out[53]:

```
<AxesSubplot:xlabel='Insulin'>
```



Here in above plot, once after the log() Tranformation because of previous data is right skewed and now median & IQR is changed and it shows more outliers on upper and lower bounds.

Because of persence of more Outliers in data, it is scaled with Robust Scaler

In [54]:

```
rs = RobustScaler(with_centering=True,
                  with_scaling=True,
                  quantile_range=(25.0, 75.0),
                  copy=True)
```

In [55]:

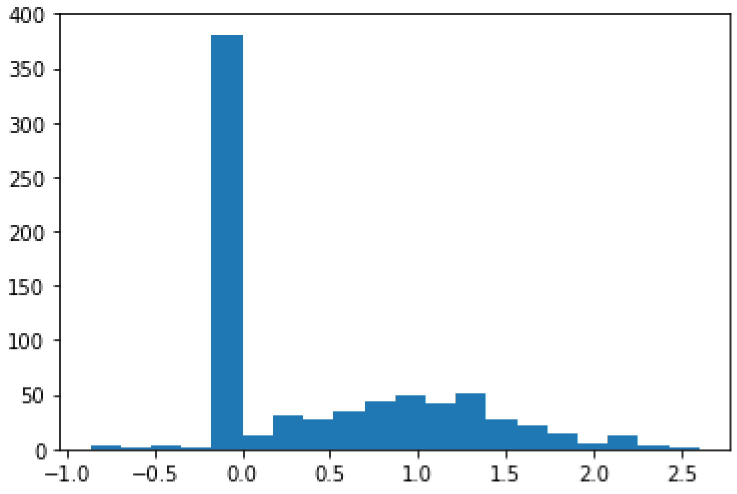
```
df['Insulin'] = rs.fit_transform(df['Insulin'].values.reshape(-1,1))
```

In [56]:

```
df.Insulin.hist(bins=20,grid=False)
```

Out[56]:

```
<AxesSubplot:>
```

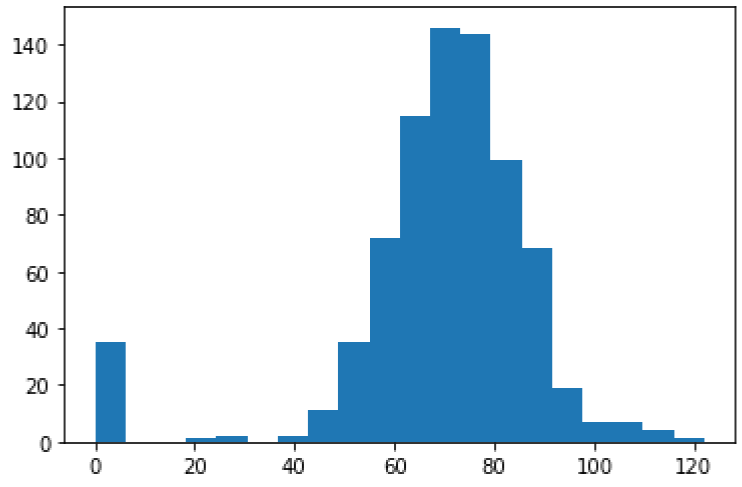


In [57]:

```
df.BloodPressure.hist(bins=20,grid=False)
```

Out[57]:

```
<AxesSubplot:>
```



In [58]:

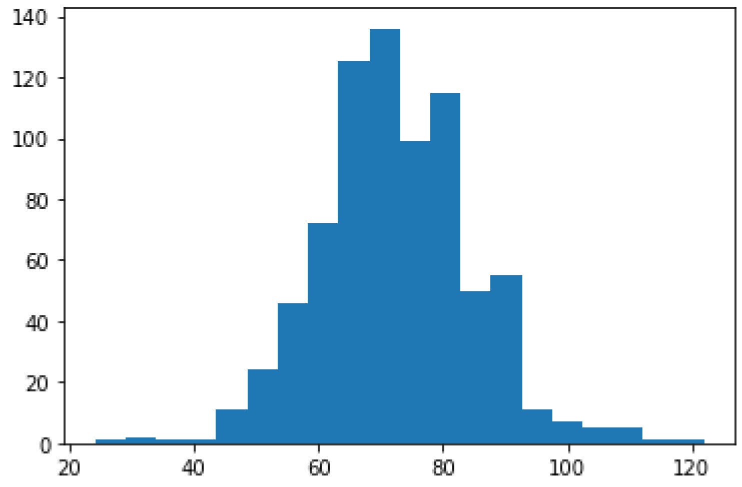
```
avg_class_0 = df.loc[class_0,'BloodPressure'].median()
avg_class_1 = df.loc[class_1,'BloodPressure'].median()
df.loc[df['BloodPressure']==0 & class_0, 'BloodPressure'] = avg_class_0
df.loc[df['BloodPressure']==0 & class_1, 'BloodPressure'] = avg_class_1
```

In [59]:

```
df.BloodPressure.hist(bins=20,grid=False)
```

Out[59]:

```
<AxesSubplot:>
```



In [60]:

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

In [61]:

```
std_scale = StandardScaler(copy=True,
                           with_mean=True,
                           with_std=True)
```

In [62]:

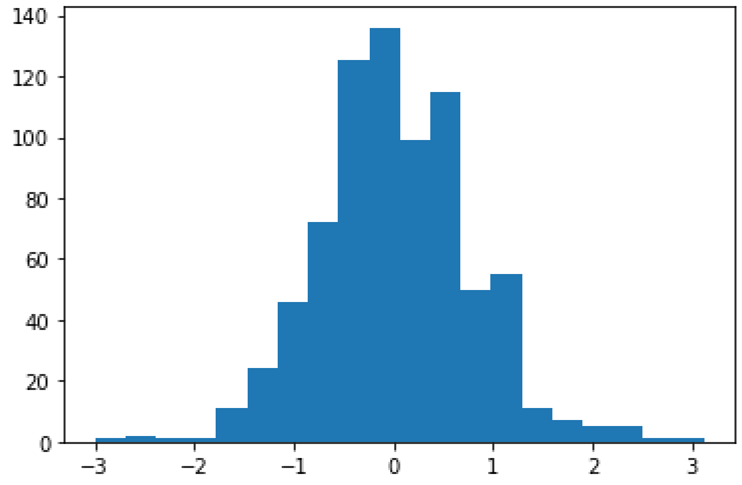
```
df['BloodPressure'] = rs.fit_transform(df['BloodPressure'].values.reshape(-1,1))
```

In [63]:

```
df.BloodPressure.hist(bins=20,grid=False)
```

Out[63]:

```
<AxesSubplot:>
```

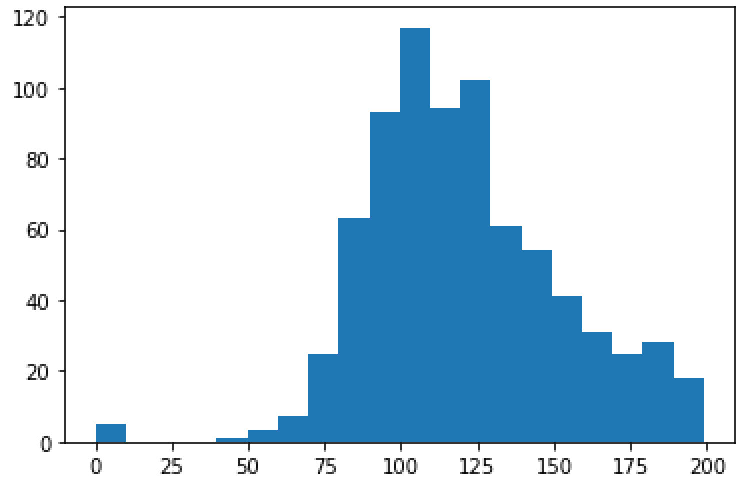


In [64]:

```
df.Glucose.hist(bins=20,grid=False)
```

Out[64]:

```
<AxesSubplot:>
```



In [65]:

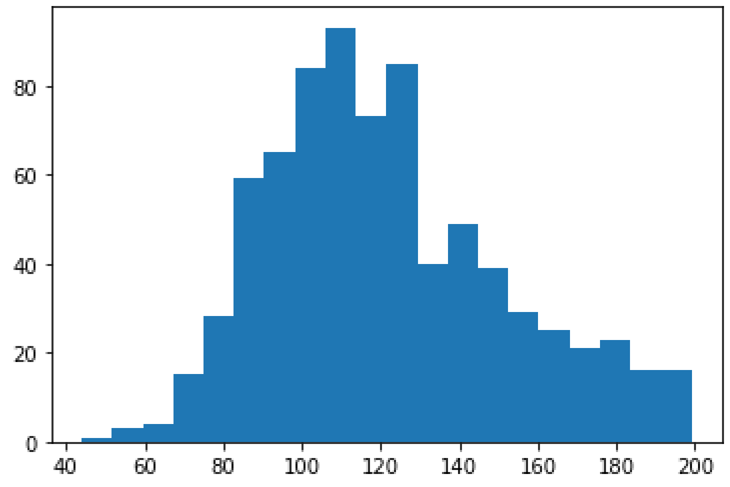
```
avg_class_0 = df.loc[class_0,'Glucose'].median()
avg_class_1 = df.loc[class_1,'Glucose'].median()
df.loc[df['Glucose']==0 & class_0, 'Glucose'] = avg_class_0
df.loc[df['Glucose']==0 & class_1, 'Glucose'] = avg_class_1
```

In [66]:

```
df.Glucose.hist(bins=20,grid=False)
```

Out[66]:

```
<AxesSubplot:>
```



In [67]:

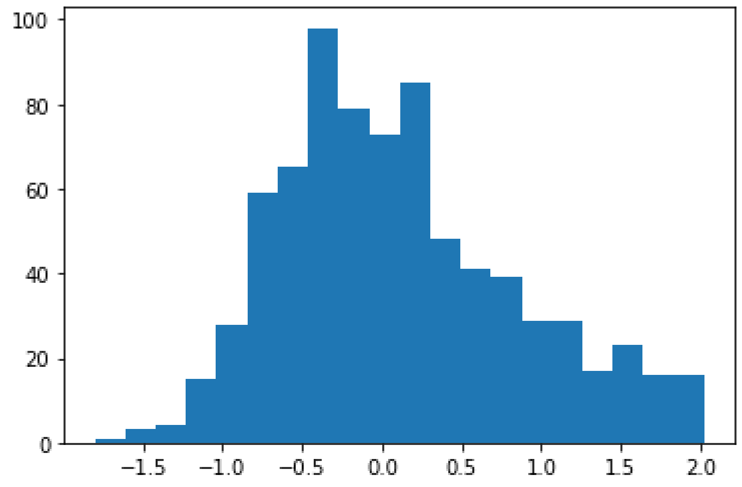
```
df['Glucose'] = rs.fit_transform(df['Glucose'].values.reshape(-1,1))
```

In [68]:

```
df.Glucose.hist(bins=20,grid=False)
```

Out[68]:

```
<AxesSubplot:>
```

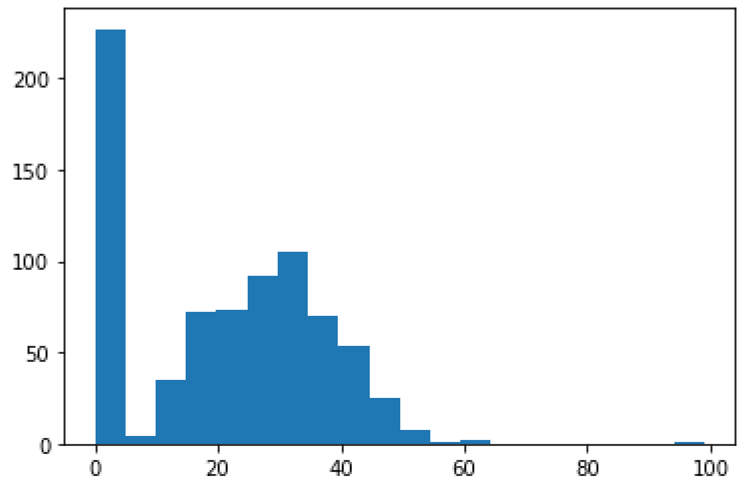


In [69]:

```
df.SkinThickness.hist(bins=20,grid=False)
```

Out[69]:

```
<AxesSubplot:>
```



In [70]:

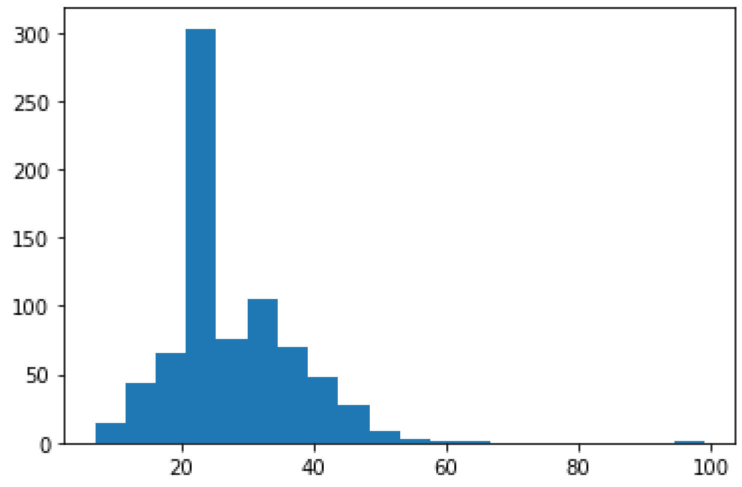
```
avg_class_0 = df.loc[class_0,'SkinThickness'].median()
avg_class_1 = df.loc[class_1,'SkinThickness'].median()
df.loc[df['SkinThickness']==0 & class_0, 'SkinThickness'] = avg_class_0
df.loc[df['SkinThickness']==0 & class_1, 'SkinThickness'] = avg_class_1
```

In [71]:

```
df.SkinThickness.hist(bins=20,grid=False)
```

Out[71]:

```
<AxesSubplot:>
```

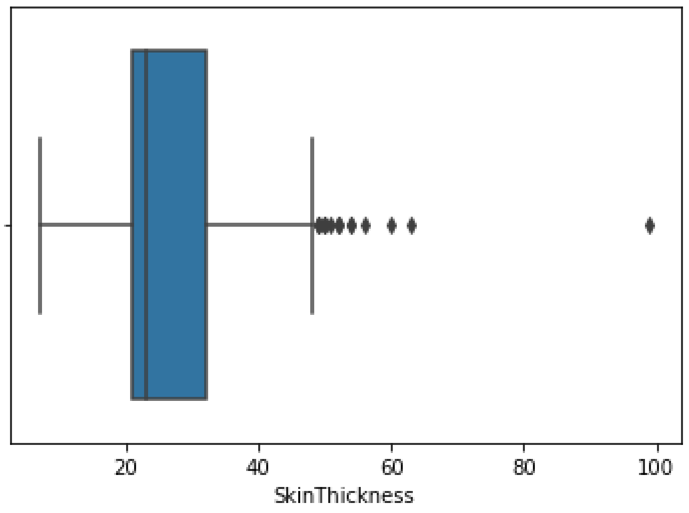


In [72]:

```
sns.boxplot(x=df.SkinThickness)
```

Out[72]:

```
<AxesSubplot:xlabel='SkinThickness'>
```



Since it has more outlier, so it is treated with Robust scaler

In [73]:

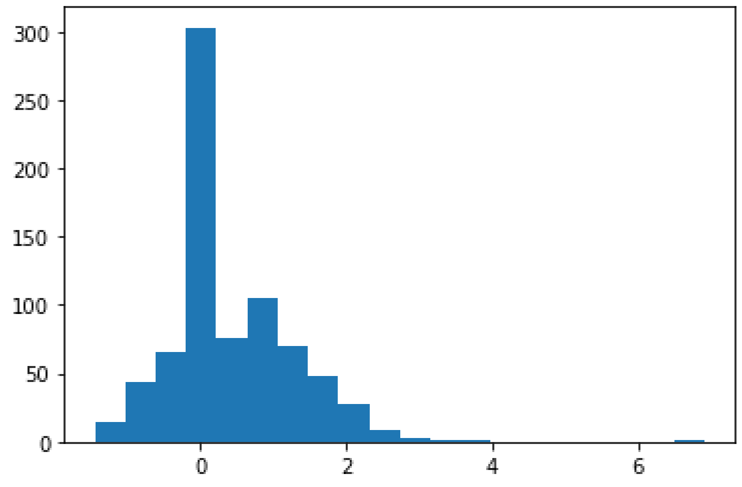
```
df['SkinThickness'] = rs.fit_transform(df['SkinThickness'].values.reshape(-1,1))
```

In [74]:

```
df.SkinThickness.hist(bins=20,grid=False)
```

Out[74]:

```
<AxesSubplot:>
```



In [75]:

```
df.head(9)
```

Out[75]:

	BMI_binned	Aged_bin	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	DiabetesPedigreeFunction	Outcome
0	2	1	6	0.765432	0.000	1.090909	0.000000	0.627	1
1	1	1	1	-0.790123	-0.375	0.545455	0.000000	0.351	0
2	0	1	8	1.629630	-0.500	-0.181818	0.000000	0.672	1
3	1	0	1	-0.691358	-0.375	0.000000	0.743906	0.167	0
4	2	1	0	0.493827	-2.000	1.090909	1.234922	2.288	1
5	1	0	5	-0.024691	0.125	-0.181818	0.000000	0.201	0
6	2	0	3	-0.962963	-1.375	0.818182	0.688132	0.248	1
7	2	0	10	-0.049383	-0.125	-0.181818	0.000000	0.134	0
8	2	2	2	1.975309	-0.125	2.000000	2.226939	0.158	1

In [76]:

```
df.Outcome.value_counts()
```

Out[76]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

Here the data is Imbalanced so it is oversampled

Oversampling of Data

In [77]:

```
smt = SMOTE(sampling_strategy='auto', random_state=9,n_jobs=-1)
x = df.drop(['Outcome'],axis = 1)
y = df.Outcome
x, y = smt.fit_sample(x,y)
```

Hold-out Method

In [78]:

```
x_train,x_test, y_train, y_test =
train_test_split(x,y,test_size=0.2,random_state=9,stratify=y)
```

In [79]:

```
print('Shape of x_train',x_train.shape)
print()
print('Shape of y_train',y_train.shape)
print()
print('Shape of x_test',x_test.shape)
print()
print('Shape of y_test',y_test.shape)
```

```
Shape of x_train (800, 8)

Shape of y_train (800,)

Shape of x_test (200, 8)

Shape of y test (200,)
```

Logistic Regression

In [80]:

```
lr = LogisticRegression(random_state=100,
                        n_jobs=-1,
                        penalty='l2',
                        solver='liblinear'
                       )
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)
```

In [81]:

```
print('test accuracy : ', accuracy_score(y_pred_lr,y_test))
```

```
test accuracy :  0.725
```

Decision Tree Classifier with oversampling

In [82]:

```
dtc = DecisionTreeClassifier(criterion="entropy",  # For the information gain
                             splitter="best",     # For the best split
                             random_state=9
                            )
```

In [83]:

```
dtc.fit(x_train,y_train)
y_pred_dtc = dtc.predict(x_test)
```

In [84]:

```
print('test accuracy : ', accuracy_score(y_pred_dtc,y_test))
```

```
test accuracy :  0.695
```

Random Forest Classifier with oversampling

In [85]:

```
rfc = RandomForestClassifier(max_depth=2,
                             random_state=0,
                             n_jobs=-1)
```

```
rfc.fit(x_train,y_train)
y_pred_rfc = dtc.predict(x_test)
print('test accuracy : ', accuracy_score(y_pred_rfc,y_test))
```

```
test accuracy :  0.695
```

Without Oversampling

In [86]:

```
x_s = df.drop('Outcome',axis=1)
y_s = df.Outcome
```

In [87]:

```
x_train,x_test, y_train, y_test = train_test_split(x_s,y_s,test_size=0.2,random_state=9)
```

In [88]:

```
print('Shape of x_train',x_train.shape)
print()
print('Shape of y_train',y_train.shape)
print()
print('Shape of x_test',x_test.shape)
print()
print('Shape of y_test',y_test.shape)
```

```
Shape of x_train (614, 8)

Shape of y_train (614,)

Shape of x_test (154, 8)

Shape of y test (154,)
```

Decision Tree Classifier

In [89]:

```
dtc1 = DecisionTreeClassifier(criterion="entropy",  # For the information gain
                              splitter="best",    # For the best split
                              random_state=9
                             )
dtc1.fit(x_train,y_train)
y_pred_dtc1 = dtc1.predict(x_test)
```

In [90]:

```
print('test accuracy : ', accuracy_score(y_pred_dtc1,y_test))
```

```
test accuracy :  0.6883116883116883
```

Random Forest Classifier

In [91]:

```
rfc1 = RandomForestClassifier(max_depth=2,
                              random_state=0,
                              n_jobs=-1)
```

In [92]:

```
rfc1.fit(x_train,y_train)
y_pred_rfc1 = rfc1.predict(x_test)
```

In [93]:

```
print('test accuracy : ', accuracy_score(y_pred_rfc1,y_test))
```

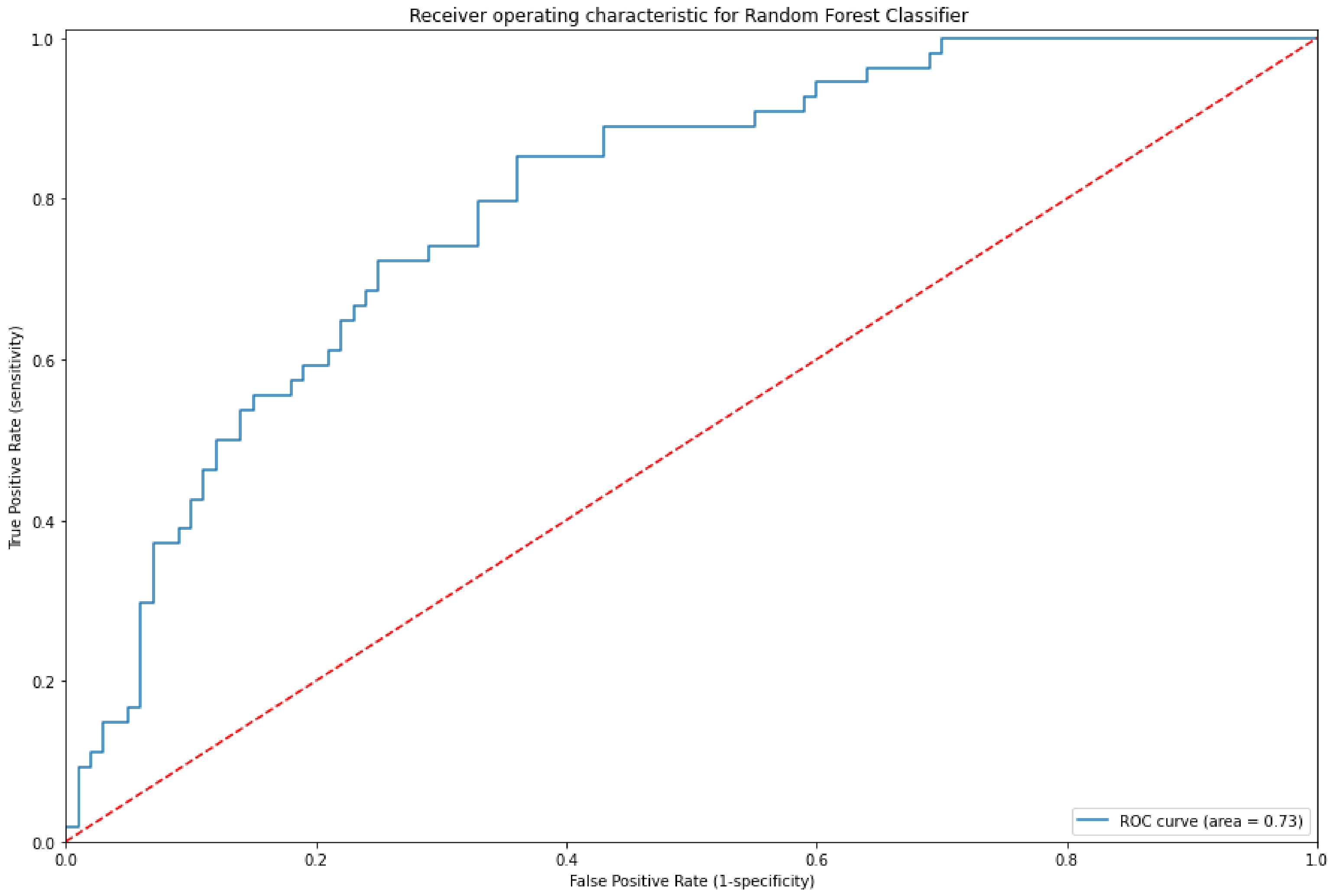
```
test accuracy :  0.7272727272727273
```

ROC AUC curve for Random Forest Classifier

In [94]:

```
fpr, tpr, thershold = roc_curve(y_test, rfc1.predict_proba(x_test)[:,1])
rfc_roc = roc_auc_score(y_pred_rfc1,y_test)
plt.figure()
plt.subplots(figsize=(15,10))
plt.plot(fpr, tpr, label = 'ROC curve (area = %0.2f)'%rfc_roc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0,1.0])
plt.ylim([0,1.01])
plt.xlabel('False Positive Rate (1-specificity)')
plt.ylabel('True Positive Rate (sensitivity)')
plt.title('Receiver operating characteristic for Random Forest Classifier ')
plt.legend(loc ="lower right")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



SVM and GridSearchCV

In [95]:

```
C=np.logspace(-2,2,5)
```

```
gamma=np.logspace(-5,5,5)
kernel=['linear', 'rbf', 'sigmoid']
param_grid = dict(C=C,gamma=gamma,kernel=kernel)
```

In [96]:

```
grid = GridSearchCV(SVC(),param_grid=param_grid,n_jobs=-1)
```

In [97]:

```
grid.fit(x_train,y_train)
y_pred_grid = grid.predict(x_test)
print('Grid Search best parameter for SVC are : ',grid.best_params_)
print()
print('SVC predicted accuracy score is : ', accuracy_score (y_pred_grid,y_test))
```

```
Grid Search best parameter for SVC are :  {'C': 100.0, 'gamma': 0.0031622776601683794, 'kern
el': 'rbf'}

SVC predicted accuracy score is :  0.7272727272727273
```

Light GBM Classifier

In [98]:

```
lgbm = LGBMClassifier(boosting_type='goss',   # Gradient-based One-Side Sampling
                      n_jobs=-1,
                      objective='binary',
                      random_state=9,
                      importance_type='split'
                      )
```

In [99]:

```
lgbm.fit(x_train, y_train)
y_pred_lgbm = lgbm.predict(x_test)
print('lgbm predicted accuracy score is : ', accuracy_score (y_pred_lgbm,y_test))
```

```
lgbm predicted accuracy score is :  0.7597402597402597
```

Light GBM Classifier gives the better accuracy when compared to other models.