DIABETES PREDICTION SYSTEM

Internship By: MERISKILL

Data Analyst: Pratyakshkumar Parmar

OBJECTIVE:

To Predict whether a Patient has Diabetes based on certain diagnostic measurements included in the dataset.

Importing Required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Loading the dataset

df =pd.read_csv("diabetes.csv")
```

Features Description

Pregnancies: No. times pregnant
Glucose:Plasma glucose concentration a 2 hours in a glucose tolerance test
BloodPressure:Diastolic blood pressure(mm) Hg)
skinThickness:Triceps skin fold thickness(mm)
Insulin: 2 hour serum insulin (mu U/ml)
BMI:Body mass index(weight in KG/ (Height in m)^2)
DiabetesPedigreeFunction: Diabetes Pedigree function
Age: Age in years

Outcome: Result **Yes if "1" or No if "0"**

In [3]:
```
# Cheking random 10 entries

df.sample(10)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
94	2	142	82	18	64	24.7	0.761	21
451	2	134	70	0	0	28.9	0.542	23
54	7	150	66	42	342	34.7	0.718	42
411	1	112	72	30	176	34.4	0.528	25
361	5	158	70	0	0	29.8	0.207	63
585	1	93	56	11	0	22.5	0.417	22
302	5	77	82	41	42	35.8	0.156	35
100	1	163	72	0	0	39.0	1.222	33
630	7	114	64	0	0	27.4	0.732	34
530	2	122	60	18	106	29.8	0.717	22

In [4]:
```
# Checking number of Rows and Columns

df.shape
```

Out[4]: (768, 9)

In [5]:
```
# Checking basic info about dataset

df.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
```

In [6]:
```
# checking missing values

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

Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```
# Checking Basic stats

df.describe().T
```

Out[7]:

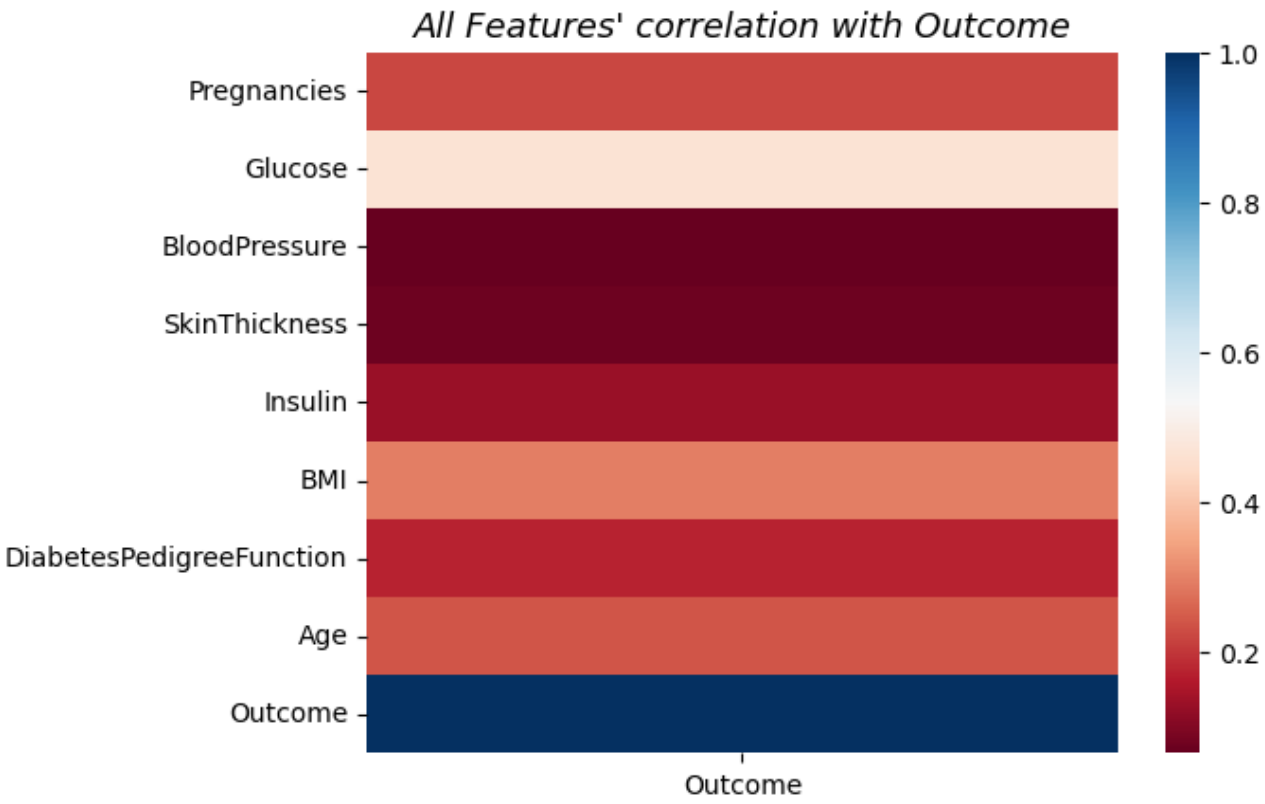
	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [8]:
```
# Finding insights aboutIndependent Features with Dependent(Outcome)

df.corr()[['Outcome']]
```

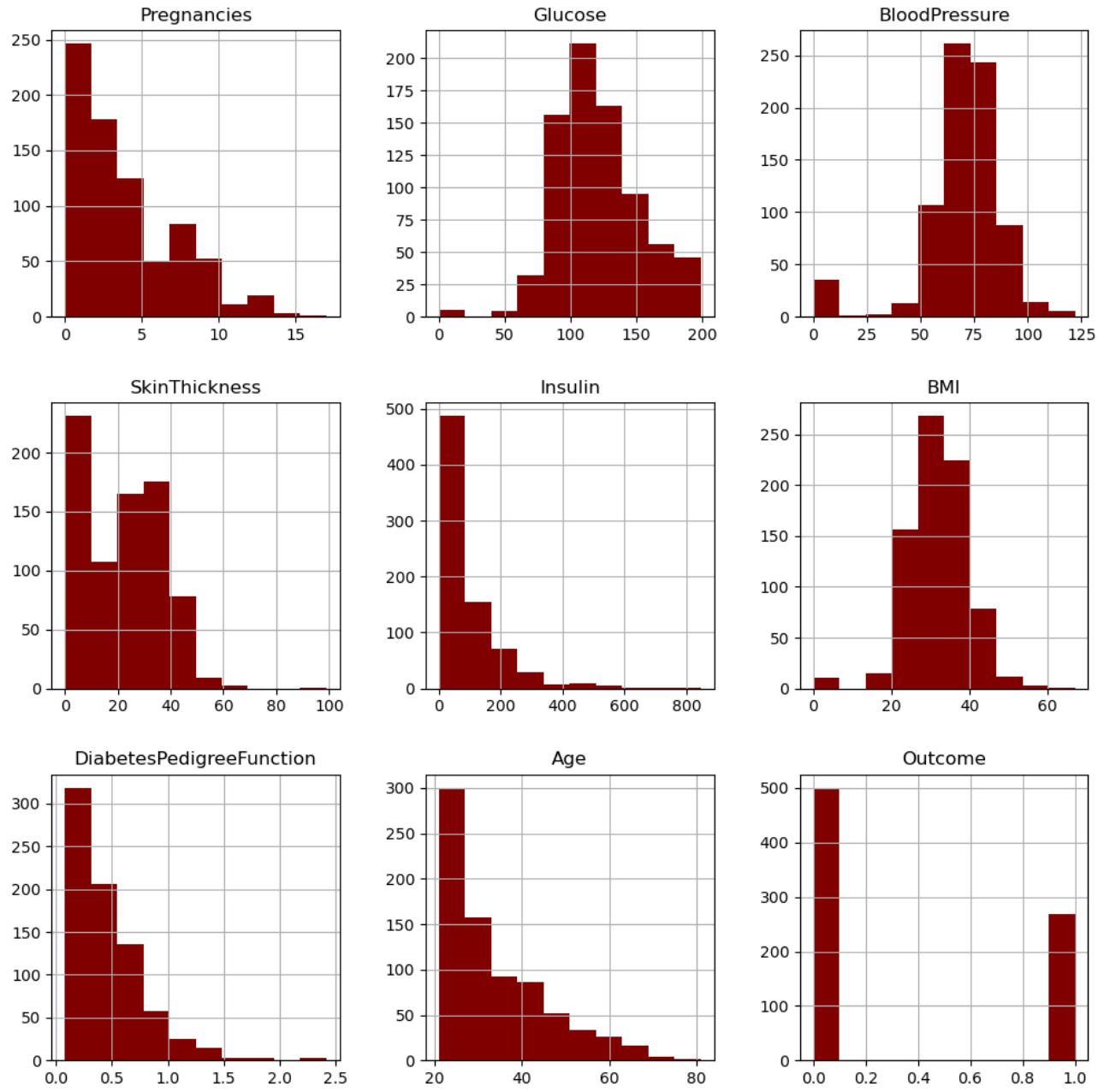
Out[8]:

	Outcome
Pregnancies	0.221898
Glucose	0.466581
BloodPressure	0.065068
SkinThickness	0.074752
Insulin	0.130548
BMI	0.292695
DiabetesPedigreeFunction	0.173844
Age	0.238356
Outcome	1.000000

In [9]:
```
# Let's check correlation using heatmap with Dependent Feature i.e. "Outcome"

plt.title("All Features' correlation with Outcome",fontsize=13,fontstyle='italic')
sns.heatmap(df.corr()[['Outcome']],cmap='RdBu')
plt.show()
```



In [10]:
```
# Checking Histplot

df.hist(figsize=(12,12),color='Maroon')
plt.show()
```



In [11]:
```
print("Minimum Values in Columns:\n")

for i in df.columns:
    x=df[i].min()
    print(i,x)
```

```
Minimum Values in Columns:

Pregnancies 0
Glucose 0
BloodPressure 0
SkinThickness 0
Insulin 0
BMI 0.0
DiabetesPedigreeFunction 0.078
Age 21
Outcome 0
```

Here, we can observe that 0 in certain columns does not make sense in real time. So,we will chnage that 0 with average value

'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI'

In [12]:
```
Glu = df.Glucose.mean()
Bld =df.BloodPressure.mean()
Ski =df.SkinThickness.mean()
Ins = df.Insulin.mean()
Bmi = df.BMI.mean()

print(f"Average of Glucose is: {Glu}")
print(f"Average of BloodPressure is: {Bld}")
print(f"Average of SkinThickness is: {Ski}")
print(f"Average of Insulin is: {Ins}")
print(f"Average of BMI is: {Bmi}")
```

```
Average of Glucose is: 120.89453125
Average of BloodPressure is: 69.10546875
Average of SkinThickness is: 20.536458333333332
Average of Insulin is: 79.79947916666667
Average of BMI is: 31.992578124999998
```

Let's make new dataframe and then modify as per requirements.

In [13]:
```
new_df =df.copy()
```

In [14]:
```
new_df.head()
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	O
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

We will now chnage 0 with average value in respective columns.

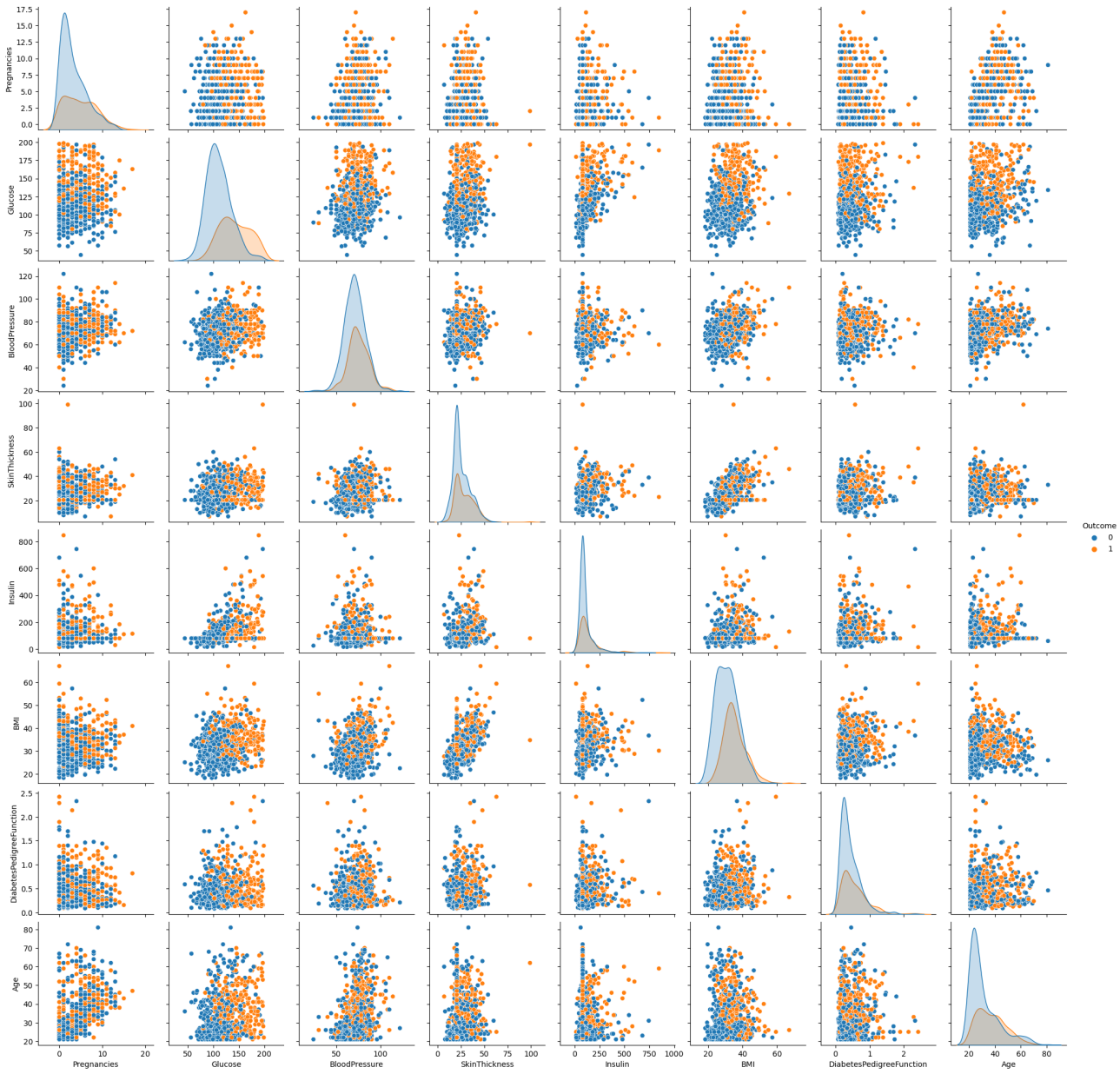
In [15]:
```
new_df['Glucose']= new_df['Glucose'].replace(0,Glu)
new_df['BloodPressure']= new_df['BloodPressure'].replace(0,Bld)
new_df['SkinThickness']= new_df['SkinThickness'].replace(0,Ski)
new_df['Insulin']= new_df['Insulin'].replace(0,Ins)
new_df['BMI']= new_df['BMI'].replace(0.00,Bmi)
```

In [16]:
```
for i in new_df.columns[1:6]:
    x=new_df[i].min()
    print(i,x)
```

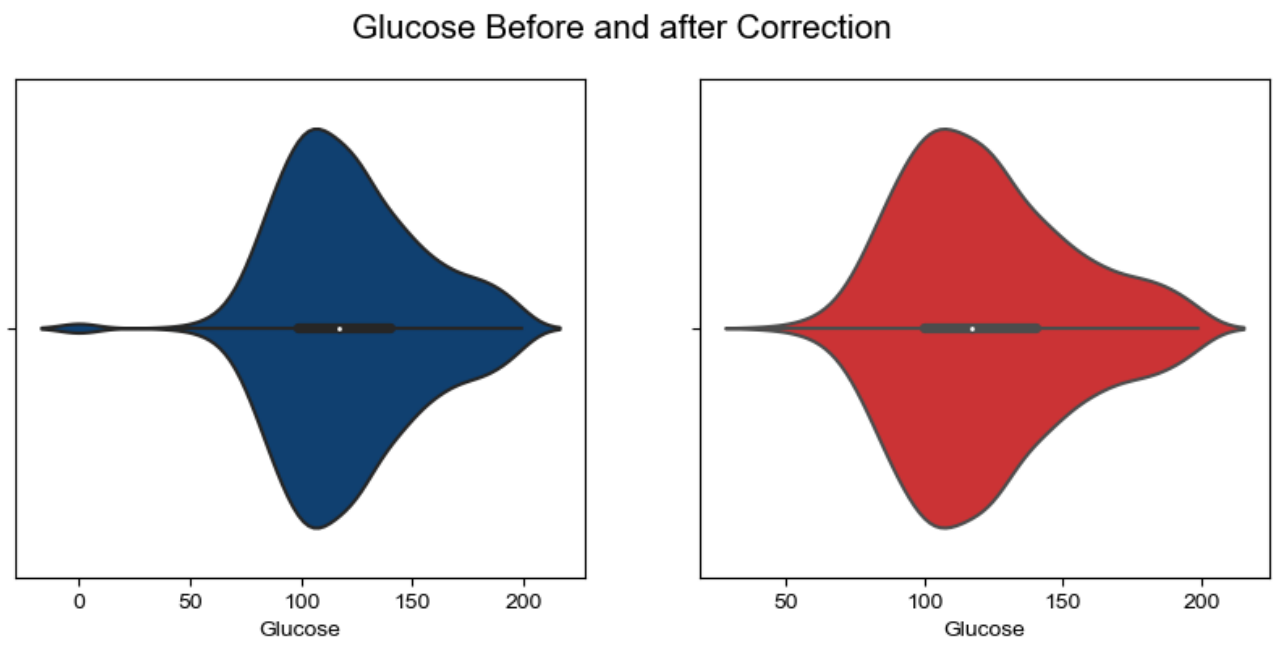
```
Glucose 44.0
BloodPressure 24.0
SkinThickness 7.0
Insulin 14.0
BMI 18.2
```

In [17]:
```
sns.pairplot(new_df,hue="Outcome")
plt.show()
```

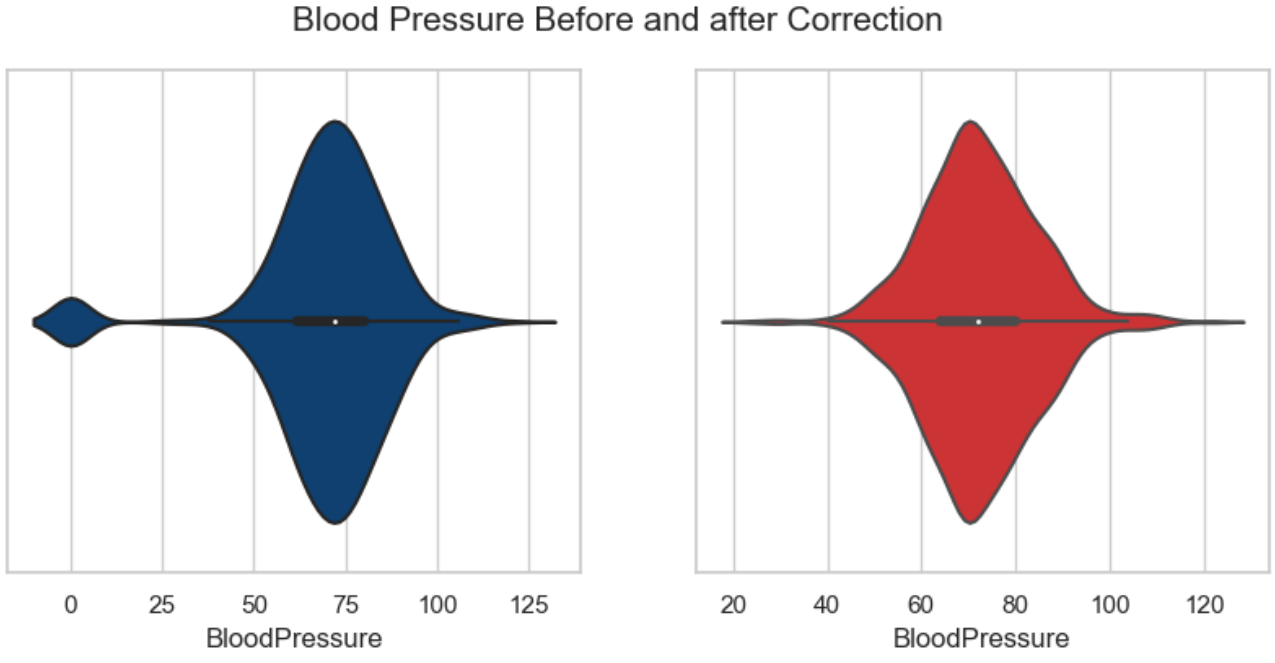
```
C:\Users\PRATYAKSH\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



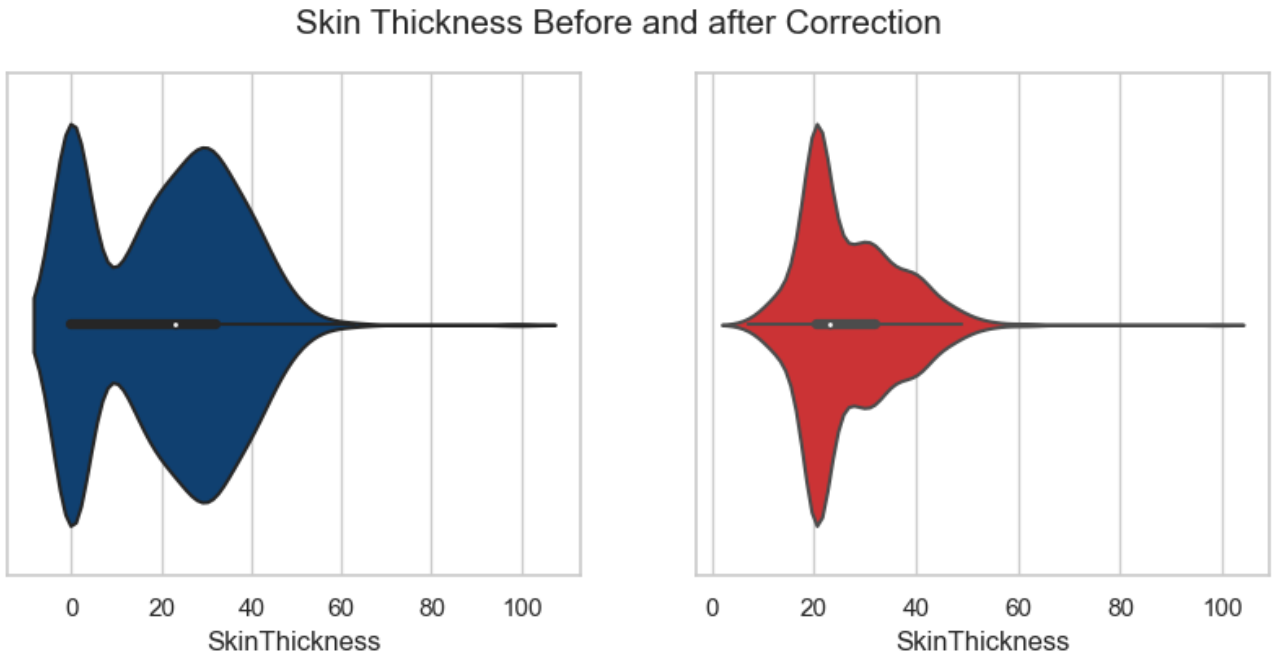
In [18]:
```
fig,axes = plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Glucose Before and after Correction",fontsize=15)

sns.set(style='whitegrid')

sns.violinplot(ax = axes[0,0], x=df.Glucose, data=df,palette='ocean')
sns.violinplot(ax = axes[0,1], x=new_df.Glucose, data=new_df,palette='Set1')

plt.show()
```



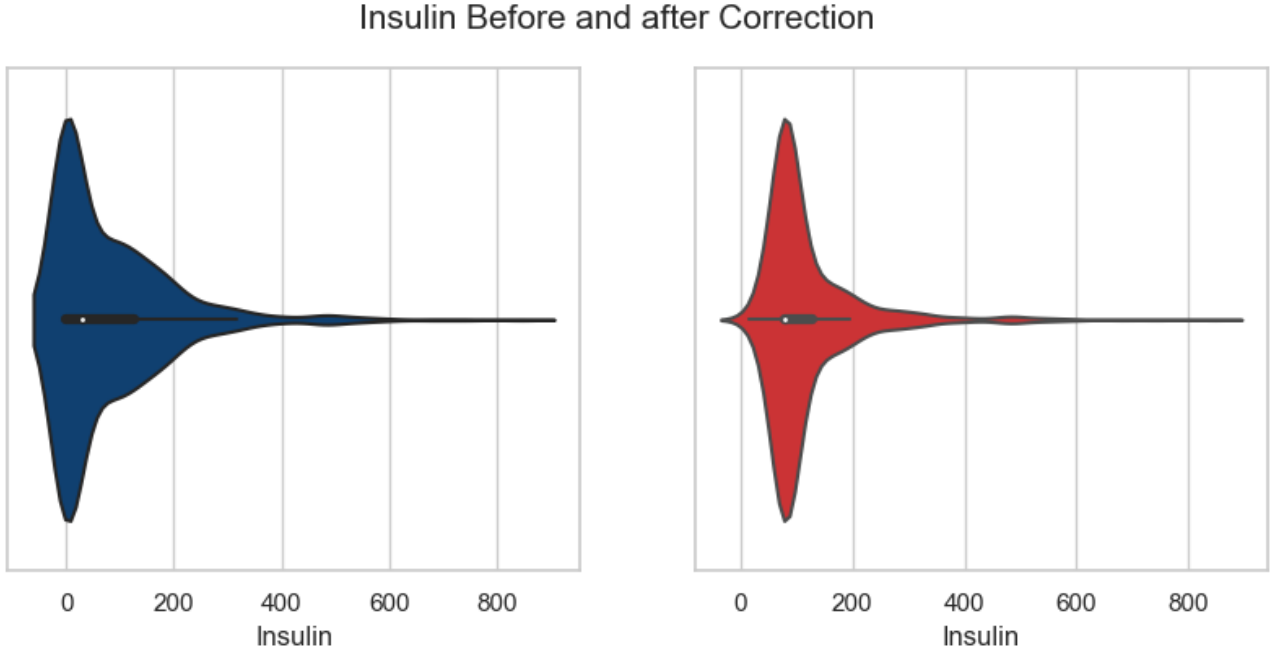
In [19]:
```
fig,axes = plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Blood Pressure Before and after Correction",fontsize=15)
sns.set(style='whitegrid')

sns.violinplot(ax = axes[0,0], x=df.BloodPressure, data=df,palette='ocean')
sns.violinplot(ax = axes[0,1], x=new_df.BloodPressure, data=new_df,palette='Set1')

plt.show()
```



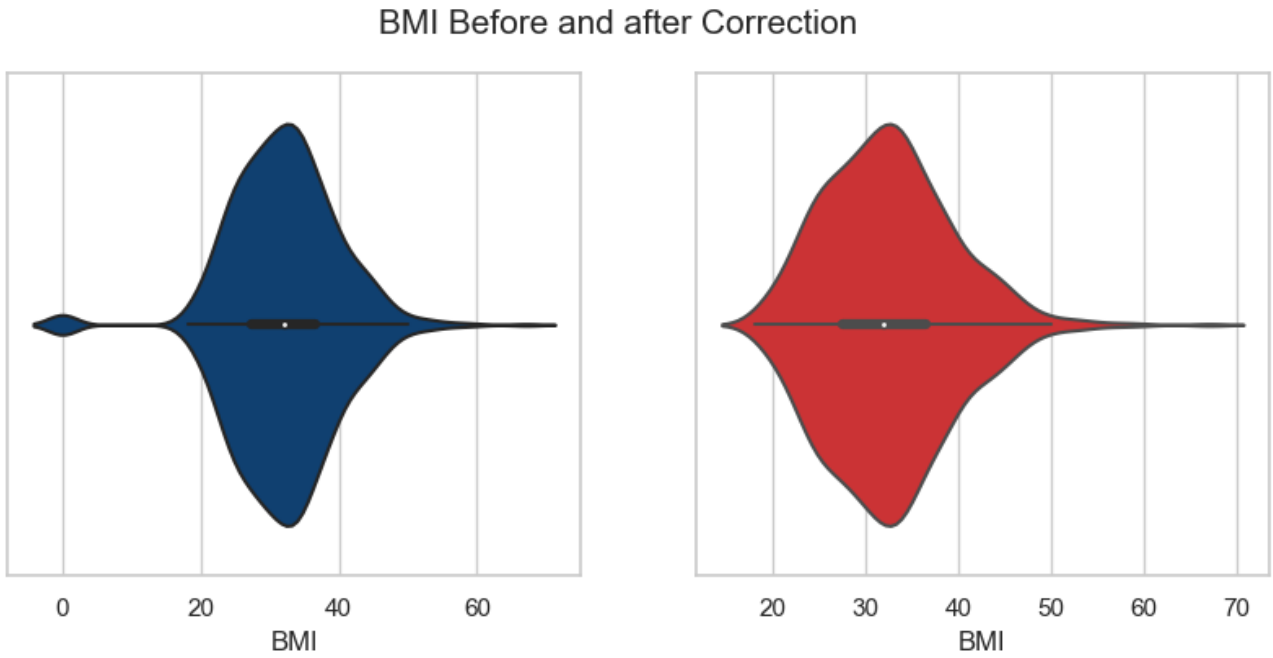
In [20]:
```
fig,axes = plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Skin Thickness Before and after Correction",fontsize=15)

sns.set(style='whitegrid')

sns.violinplot(ax = axes[0,0], x=df.SkinThickness, data=df,palette='ocean')
sns.violinplot(ax = axes[0,1], x=new_df.SkinThickness, data=new_df,palette='Set1')

plt.show()
```



In [21]:
```
fig,axes = plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Insulin Before and after Correction",fontsize=15)

sns.set(style='whitegrid')

sns.violinplot(ax = axes[0,0], x=df.Insulin, data=df,palette='ocean')
sns.violinplot(ax = axes[0,1], x=new_df.Insulin, data=new_df,palette='Set1')

plt.show()
```



In [22]:
```
fig,axes = plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("BMI Before and after Correction",fontsize=15)

sns.set(style='whitegrid')

sns.violinplot(ax = axes[0,0], x=df.BMI, data=df,palette='ocean')
sns.violinplot(ax = axes[0,1], x=new_df.BMI, data=new_df,palette='Set1')

plt.show()
```



Let's work on Machine learning model to Predict Outcome

In [23]:
```
# Features selection

X = new_df.drop('Outcome',axis=1)
y = new_df['Outcome']
```

In [24]:
```
# To split data into train and test

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2, random_state=16)
```

In [25]:
```
# Using Logistic Regression model

from sklearn.linear_model import LogisticRegression

model =LogisticRegression(random_state=16)
model.fit(X_train,y_train)
```

```
C:\Users\PRATYAKSH\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:46
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[25]:

LogisticRegression

LogisticRegression(random_state=16)

In [26]:
```
y_pred = model.predict(X_test)
```

In [27]:
```
# import the metrics class
from sklearn import metrics

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

Out[27]:
```
array([[94,  8],
       [21, 31]], dtype=int64)
```

In [28]:
```
from sklearn.metrics import classification_report

target_names = ['Diabetes Not Detected', 'Diabetes Detected']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
                      precision    recall  f1-score   support

Diabetes Not Detected       0.82      0.92      0.87       102
    Diabetes Detected       0.79      0.60      0.68        52

             accuracy                           0.81       154
            macro avg       0.81      0.76      0.77       154
         weighted avg       0.81      0.81      0.80       154
```

In [29]:
```
# Checking accuracy of the model

from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_pred,y_test)
print(f"Accuracy of the model is: {accuracy*100}")
```

```
Accuracy of the model is: 81.16883116883116
```

Thank You

GitHub Link:

https://github.com/ipratyaksh21/DIABETES-PREDICTION-FROM-MERISKILL-INTERNSHIP.git