diabetes

November 15, 2023

0.0.1 Diabetes Prediction System

Data Analysis Internship by MeriSkill

Analyst: Titiksha Agrawal

Objective:

To predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset

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

[2]:
```
data=pd.read_csv('diabetes.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]
```

Feature Description:- Pregnancies: No. of times pregnant

Glucose: Plasma glucose concentration a 2 hr in glucose tolerance test

BloodPressure: Diastlic blood pressure

SkinThickness: Triceps skin fold thickness

Insulin: 2 hr serum insulin

BMI: Body Mass Index

DiabetesPedigreeFunction: Diabetes Pedigree Fundtion

Age: Age in years

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

[4]:
```
data.head()
```

[4]:
```
   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
```

[5]:
```
data.tail()
```

[5]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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
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
```

[6]:
```
data.sample(10)
```

[6]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
35             4      103             60             33      192  24.0
652            5      123             74             40       77  34.1
207            5      162            104              0        0  37.7
707            2      127             46             21      335  34.4
432            1       80             74             11       60  30.0
563            6       99             60             19       54  26.9
326            1      122             64             32      156  35.1
545            8      186             90             35      225  34.5
552            6      114             88              0        0  27.8
690            8      107             80              0        0  24.6

     DiabetesPedigreeFunction  Age  Outcome
35                      0.966   33        0
652                     0.269   28        0
207                     0.151   52        1
707                     0.176   22        0
432                     0.527   22        0
563                     0.497   32        0
326                     0.692   30        1
545                     0.423   37        1
552                     0.247   66        0
690                     0.856   34        0
```

[7]:
```
data.shape
```

[7]:
```
(768, 9)
```

[8]:
```
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
```

[9]:
```
data.describe()
```

[9]:

	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

[10]:
```
data.isnull().sum()
```

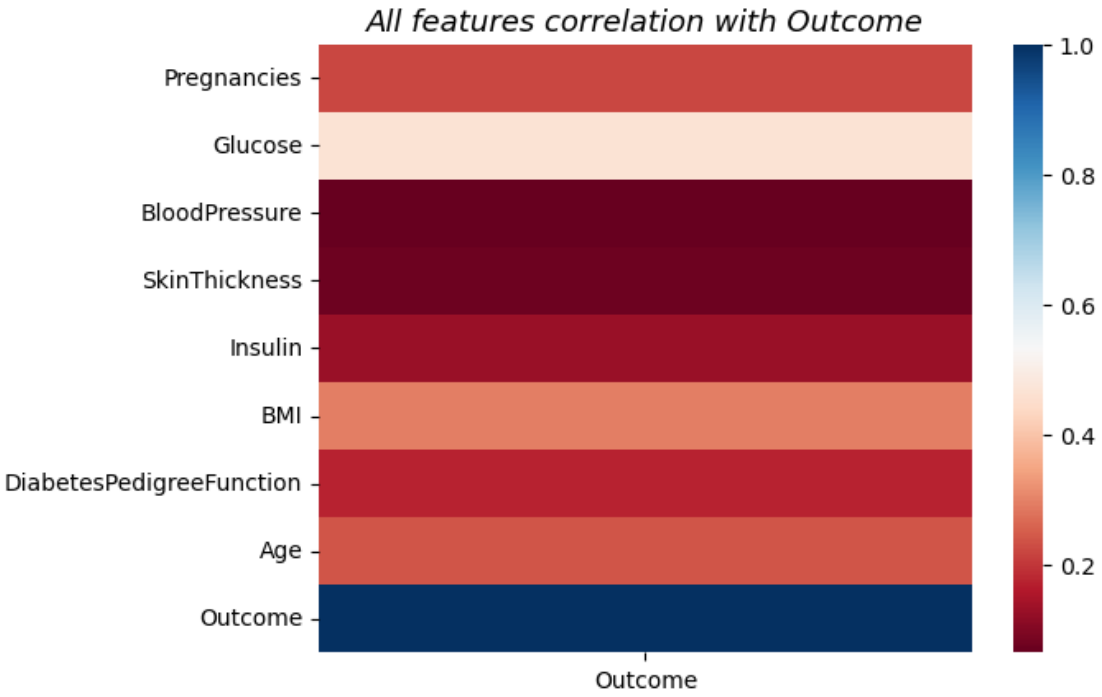
[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[11]:
```
data.corr()[['Outcome']]
```

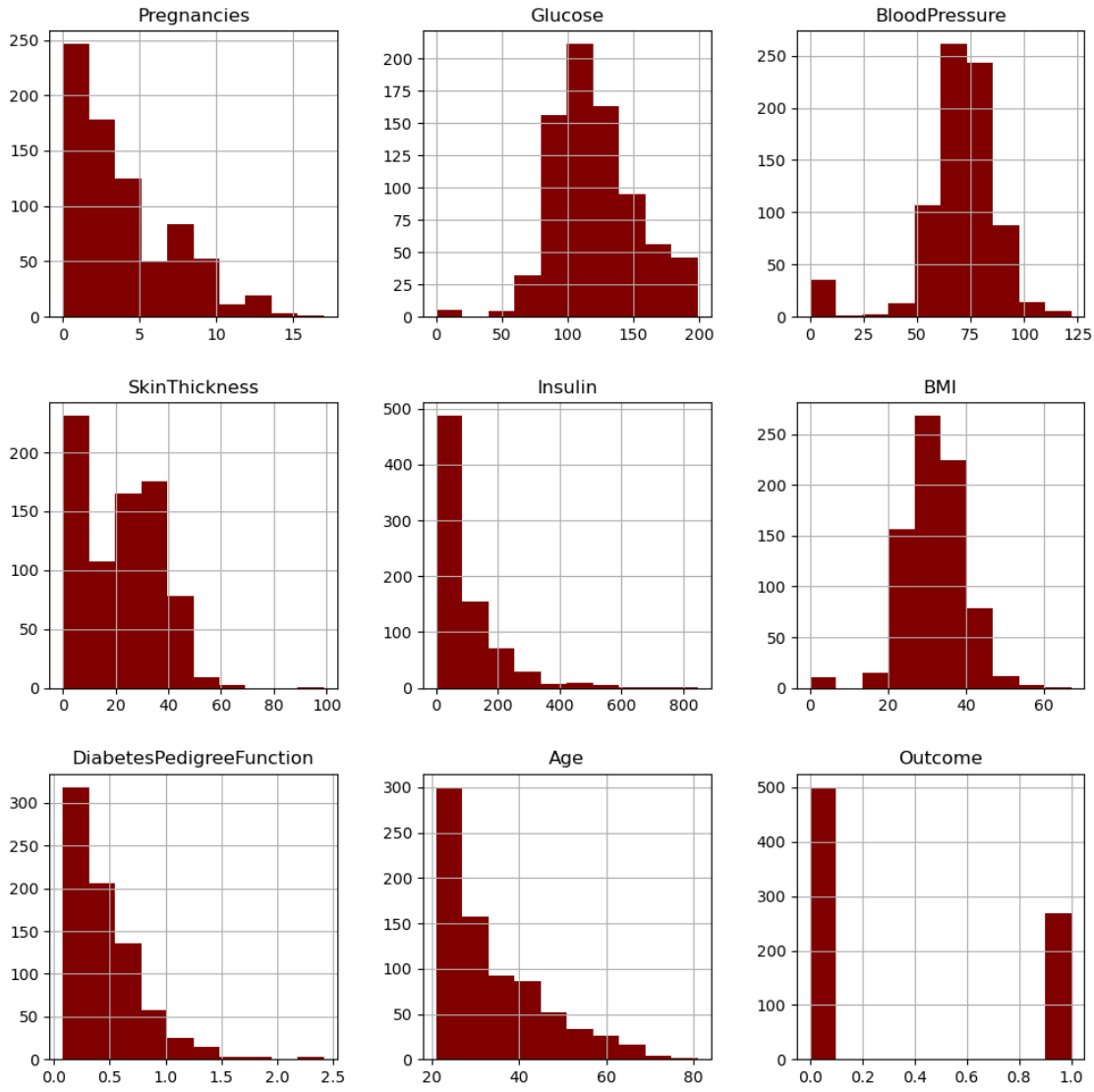
[11]:

	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

[12]:
```
plt.title("All features correlation with␣
 ↪Outcome",fontsize=13,fontstyle='italic')
sns.heatmap(data.corr()[['Outcome']],cmap='RdBu')
plt.show()
```



[13]:
```
data.hist(figsize=(12,12),color='Maroon')
plt.show()
```



[14]:
```
print("Minimum values in column:\n\n")
for i in data.columns:
    x=data[i].min()
    print(i,x)
```

```
Minimum values in column:


Pregnancies 0
Glucose 0
BloodPressure 0
SkinThickness 0
Insulin 0
```

```
BMI 0.0
DiabetesPedigreeFunction 0.078
Age 21
Outcome 0
```

Replacing minimum value (0) in the above columns with mean values : Glucose, BloodPressure, SkinThickness, Insulin, BMI. By this the data will be more meaningful

[15]:
```
glu=data.Glucose.mean()
bld=data.BloodPressure.mean()
ski=data.SkinThickness.mean()
ins=data.Insulin.mean()
bmi=data.BMI.mean()

print(f"Average of Glucose: {glu}")
print(f"Average of blood pressure: {bld}")
print(f"Average of skin thickness: {ski}")
print(f"Average of Insulin: {ins}")
print(f"Average of BMI: {bmi}")
```

```
Average of Glucose: 120.89453125
Average of blood pressure: 69.10546875
Average of skin thickness: 20.536458333333332
Average of Insulin: 79.79947916666667
Average of BMI: 31.992578124999998
```

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

[16]:
```
new_data=data.copy()
```

[17]:
```
new_data.head()
```

[17]:

	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

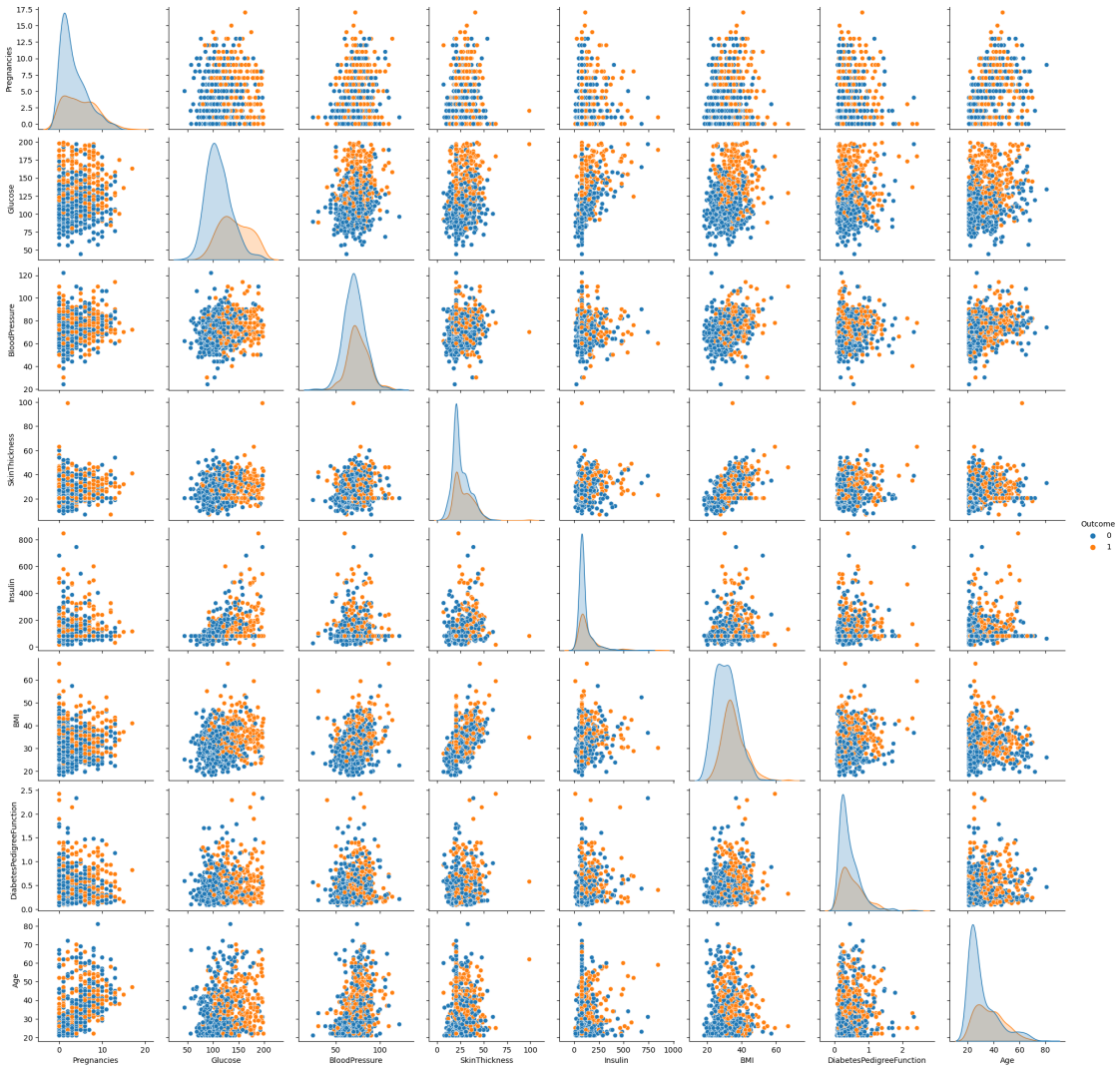
[18]:
```
new_data['Glucose']=new_data['Glucose'].replace(0,glu)
new_data['BloodPressure']=new_data['BloodPressure'].replace(0,bld)
new_data['SkinThickness']=new_data['SkinThickness'].replace(0,ski)
```

```
new_data['Insulin']=new_data['Insulin'].replace(0,ins)
new_data['BMI']=new_data['BMI'].replace(0,bmi)
```

[19]:
```
for i in new_data.columns[1:6]:
    x=new_data[i].min()
    print(i,x)
```

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

[22]:
```
sns.pairplot(new_data,hue="Outcome")
plt.show()
```

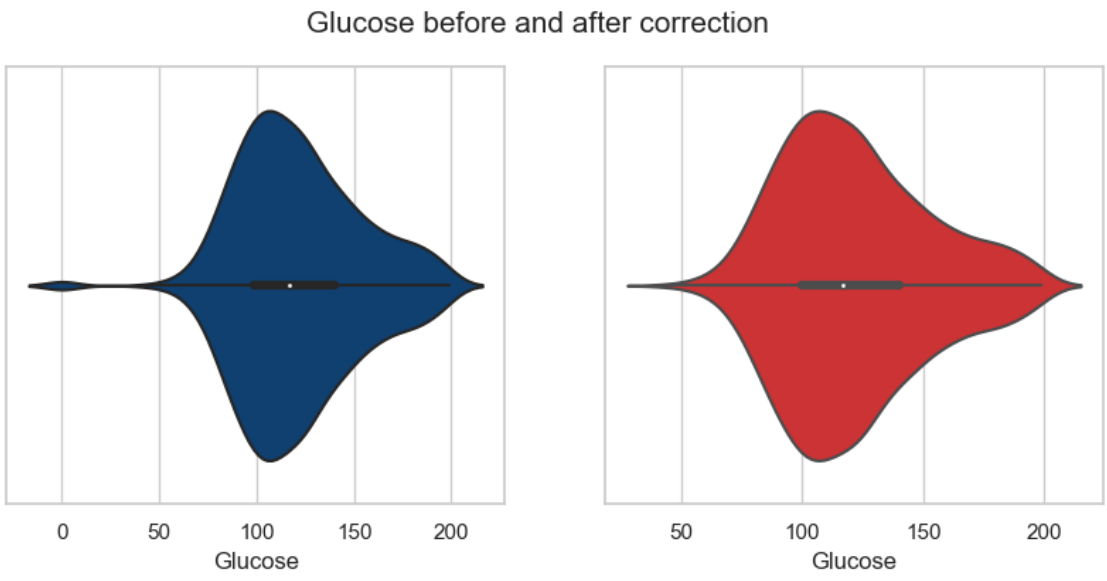


[29]:
```
fig,axes=plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Glucose before and after correction",fontsize=15)

sns.set(style="whitegrid")

sns.violinplot(ax=axes[0,0],x=data.Glucose, data1=data,palette='ocean')
sns.violinplot(ax=axes[0,1],x=new_data.Glucose, data1=new_data,palette='Set1')

plt.show()
```

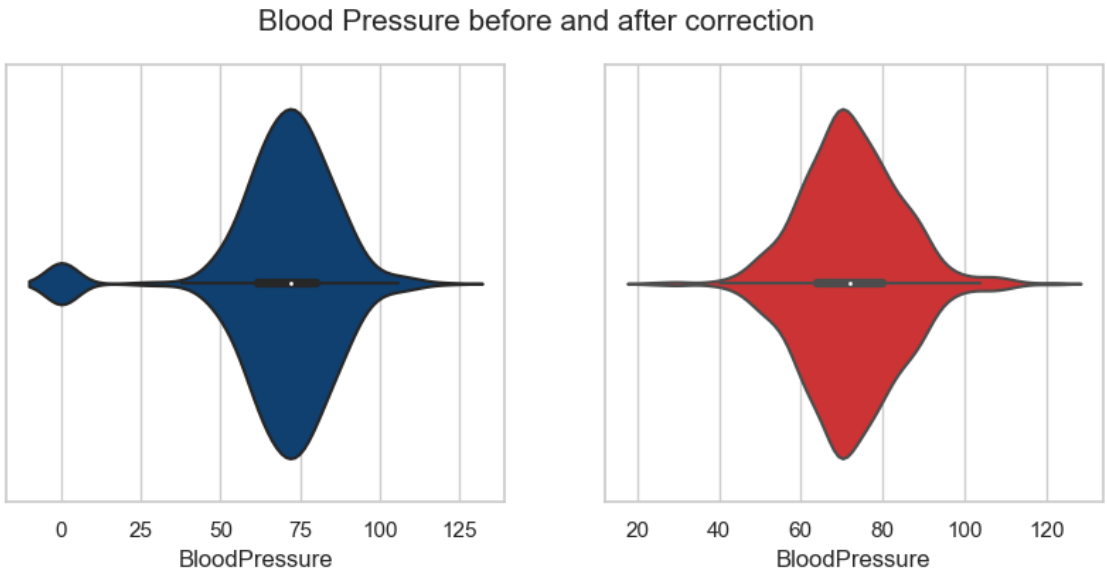


[32]:
```
fig,axes=plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Blood Pressure before and after correction",fontsize=15)

sns.set(style="whitegrid")

sns.violinplot(ax=axes[0,0],x=data.BloodPressure, data1=data,palette='ocean')
sns.violinplot(ax=axes[0,1],x=new_data.BloodPressure,␣
 ↪data1=new_data,palette='Set1')

plt.show()
```

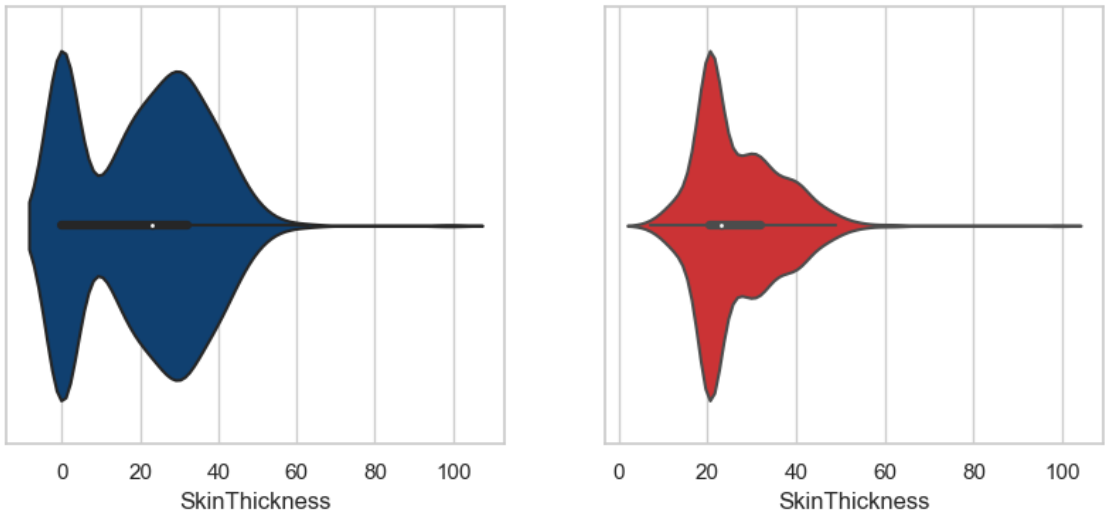


[33]:
```
fig,axes=plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Skin Thickness before and after correction",fontsize=15)

sns.set(style="whitegrid")

sns.violinplot(ax=axes[0,0],x=data.SkinThickness, data1=data,palette='ocean')
sns.violinplot(ax=axes[0,1],x=new_data.SkinThickness,␣
 ↪data1=new_data,palette='Set1')

plt.show()
```

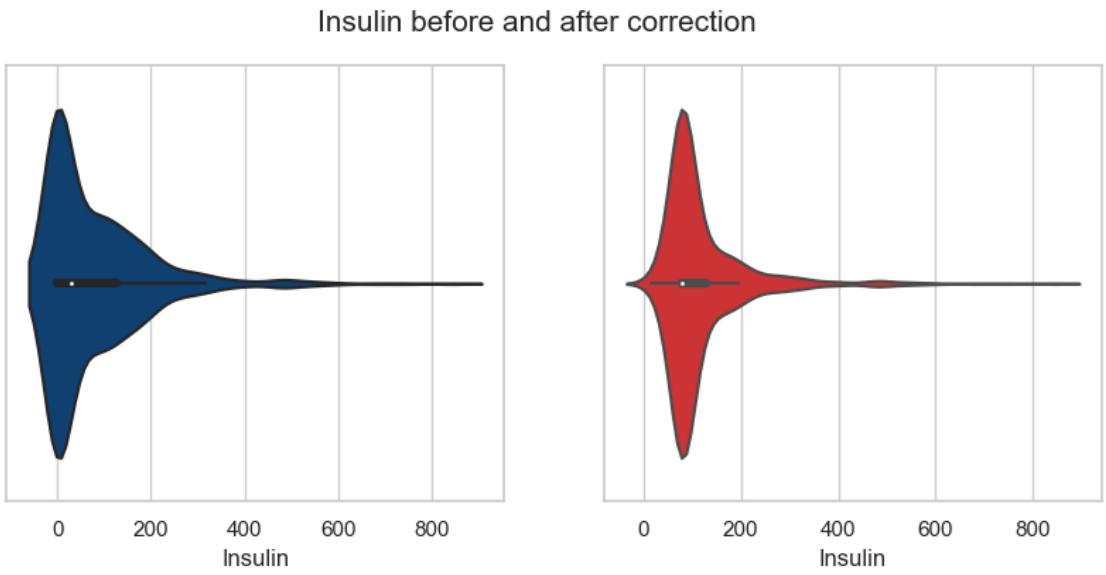


[34]:
```
fig,axes=plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("Insulin before and after correction",fontsize=15)

sns.set(style="whitegrid")

sns.violinplot(ax=axes[0,0],x=data.Insulin, data1=data,palette='ocean')
sns.violinplot(ax=axes[0,1],x=new_data.Insulin, data1=new_data,palette='Set1')

plt.show()
```

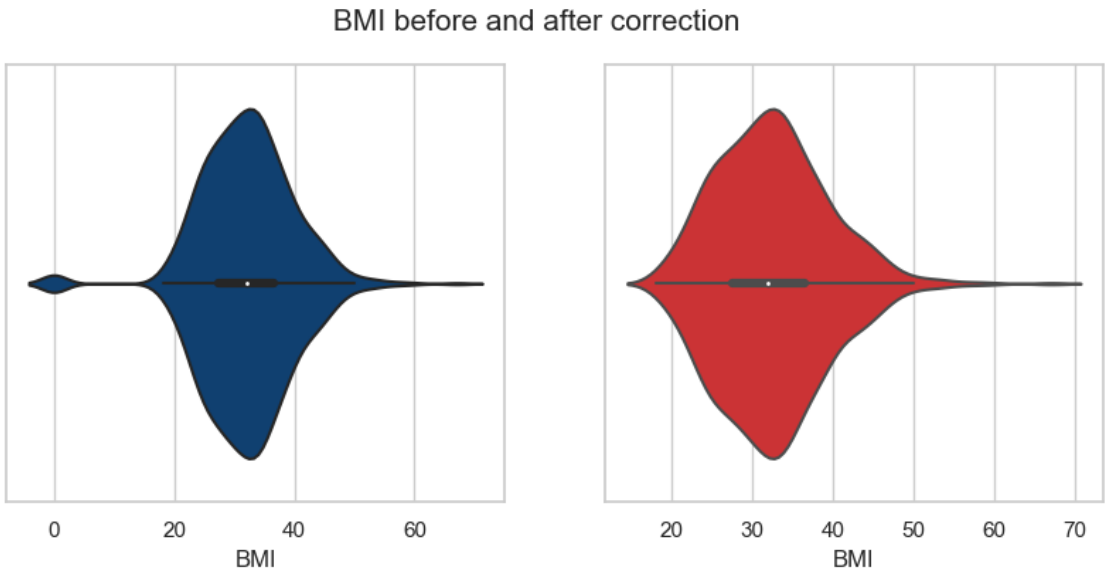


[35]:
```
fig,axes=plt.subplots(1,2,figsize=(10,4),squeeze=False)
plt.suptitle("BMI before and after correction",fontsize=15)

sns.set(style="whitegrid")

sns.violinplot(ax=axes[0,0],x=data.BMI, data1=data,palette='ocean')
sns.violinplot(ax=axes[0,1],x=new_data.BMI, data1=new_data,palette='Set1')

plt.show()
```



[36]:
```
x=new_data.drop('Outcome',axis=1)
y=new_data['Outcome']
```

[37]:
```
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)
```

[38]:
```
from sklearn.linear_model import LogisticRegression

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

```
C:\Users\titik\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:458: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[38]:
```
LogisticRegression(random_state=16)
```

[39]:
```
y_pred=model.predict(x_test)
```

[40]:
```
from sklearn import metrics

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

[40]:
```
array([[94,  8],
       [21, 31]], dtype=int64)
```

[41]:
```
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
```

[45]:
```
from sklearn.metrics import accuracy_score

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

```
Accuracy of model: 81.16883116883116
```