MeriSkill Internship Program

Name: Sannidhya Das

Task 2: Diabetics Prediction

Description:

The objective of the dataset is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Import libraries and Dataset

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

importing the data

In [2]:
```
data=pd.read_csv("diabetes.csv")
data
```

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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

viewing top 5 rows

In [3]:
```
data.head()
```

Out[3]:

	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

viewing the dimensions of the dataset

In [4]:
```
data.shape
```
Out[4]:
```
(768, 9)
```

viewing dataframe summary

In [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
```

viewing descriptive statistics

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [7]:
```
data.columns
```
Out[7]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Checking for null values

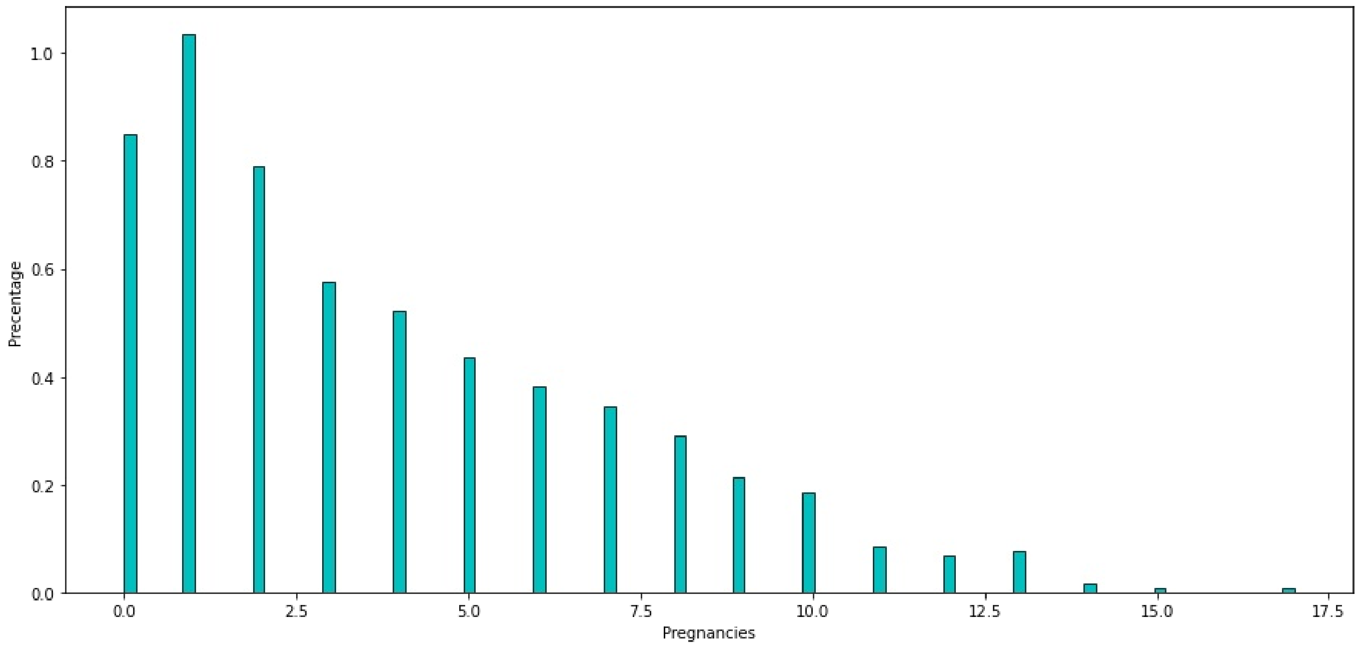
In [8]:
```
data.isnull().sum()
```
Out[8]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Exploratory Data Analysis

In [9]:
```
#Pregnencies variable distn

plt.figure(figsize=(15,7))
sns.histplot(data['Pregnancies'], facecolor='c',bins=100,stat='density')
plt.ylabel('Precentage')
plt.xlabel('Pregnancies')
```

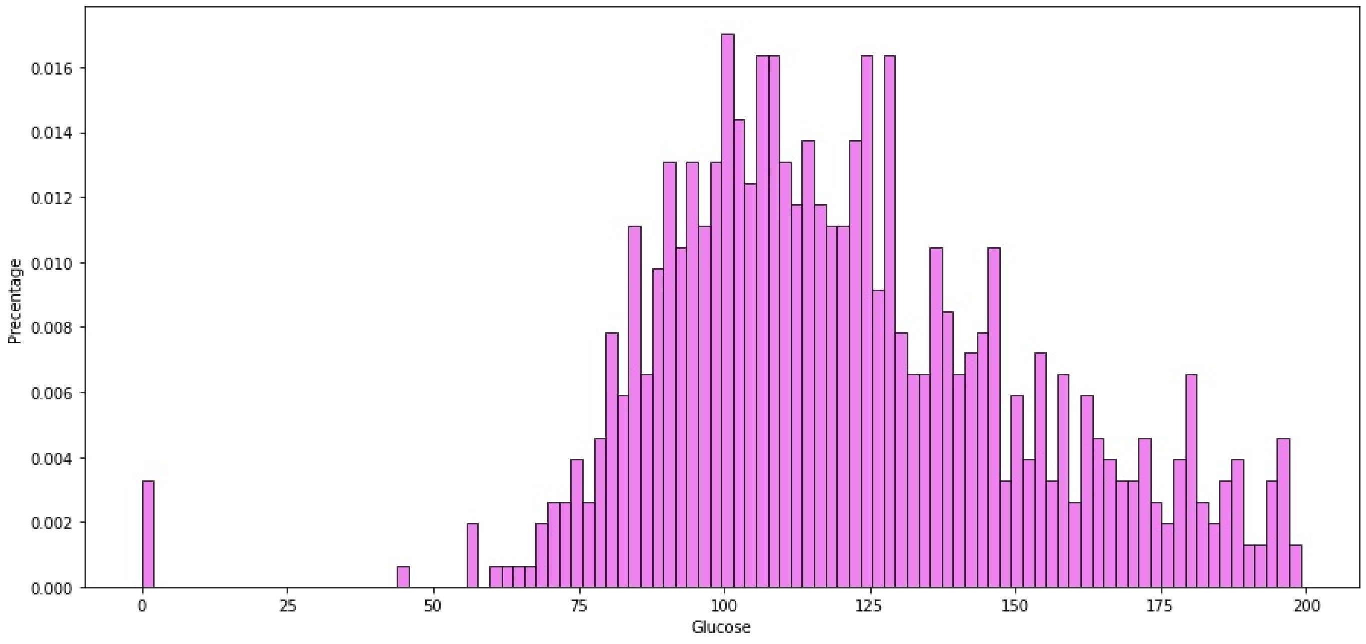
Out[9]:
```
Text(0.5, 0, 'Pregnancies')
```



In [10]:
```
#Glucose variable distn

plt.figure(figsize=(15,7))
sns.histplot(data['Glucose'], facecolor='violet',bins=100,stat='density')
plt.ylabel('Precentage')
plt.xlabel('Glucose')
```

Out[10]:
```
Text(0.5, 0, 'Glucose')
```

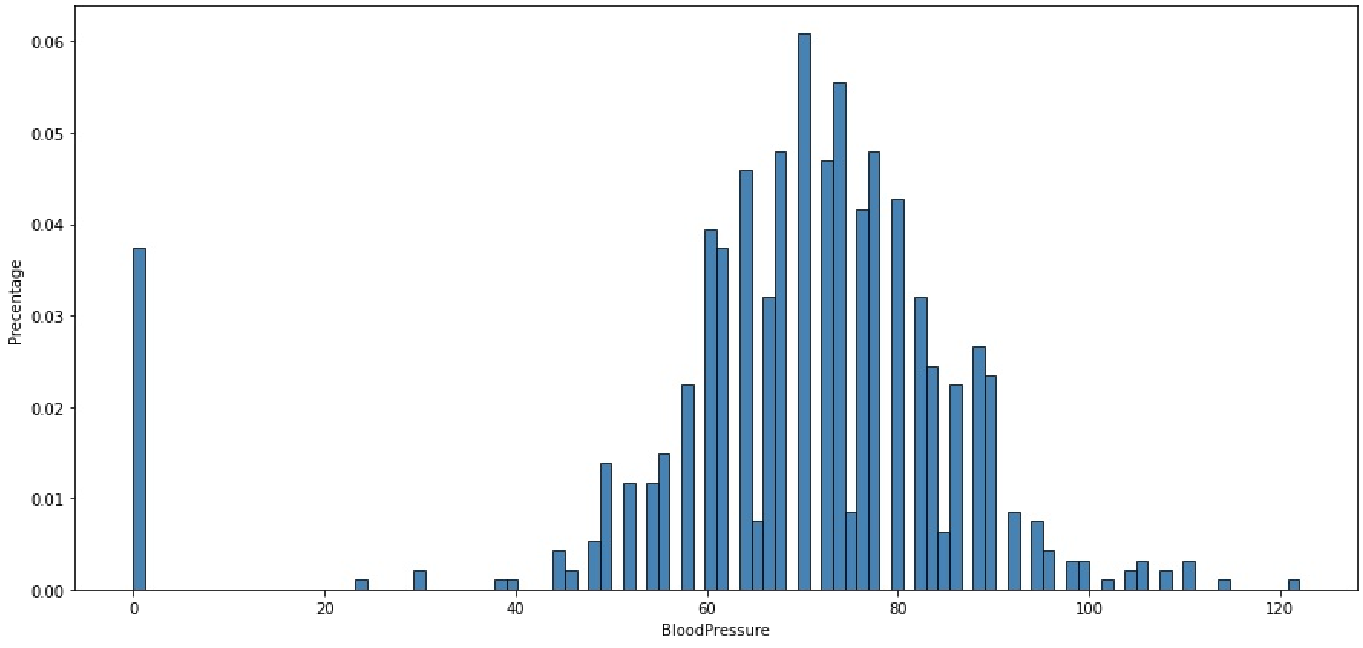


In [11]:
```
#Blood Pressure distn

plt.figure(figsize=(15,7))
sns.histplot(data['BloodPressure'], facecolor='steelblue',bins=100,stat='density')
```

```
plt.ylabel('Precentage')
plt.xlabel('BloodPressure')
```

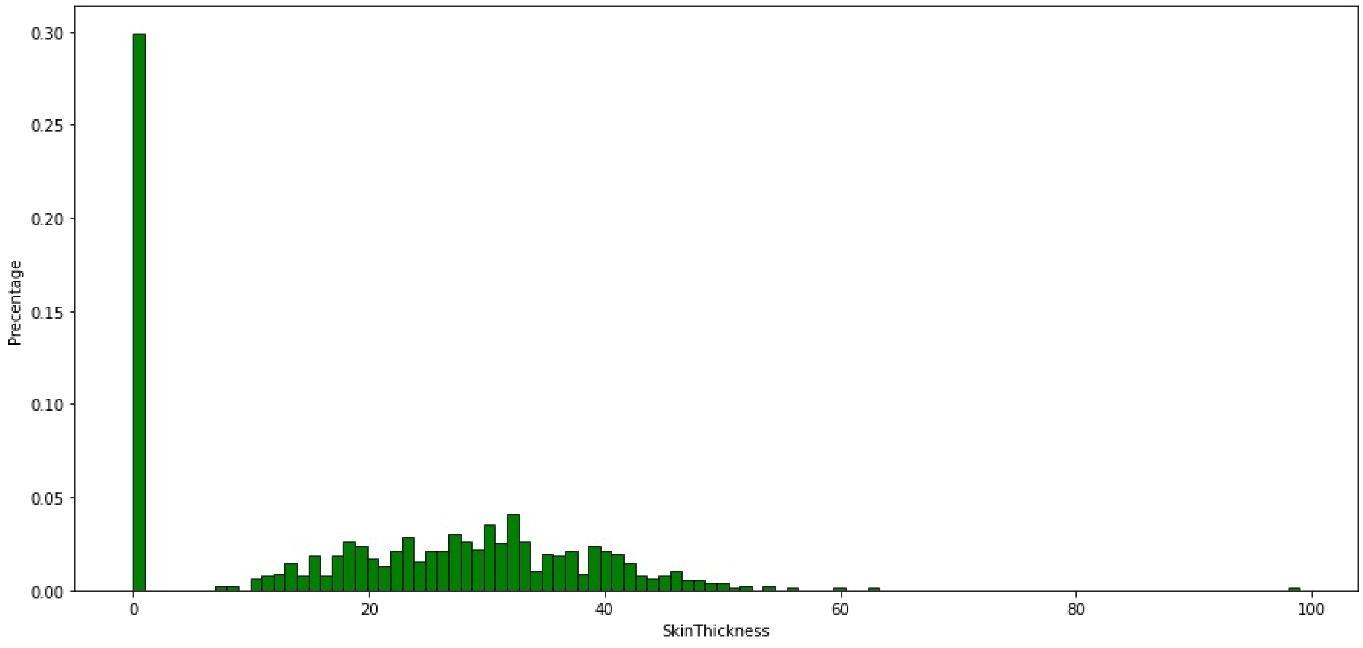
Out[11]:
```
Text(0.5, 0, 'BloodPressure')
```



In [12]:
```
#Skin Sickness distn

plt.figure(figsize=(15,7))
sns.histplot(data['SkinThickness'], facecolor='green',bins=100,stat='density')
plt.ylabel('Precentage')
plt.xlabel('SkinThickness')
```

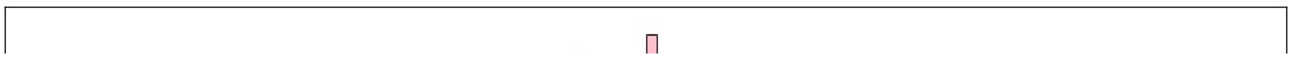
Out[12]:
```
Text(0.5, 0, 'SkinThickness')
```

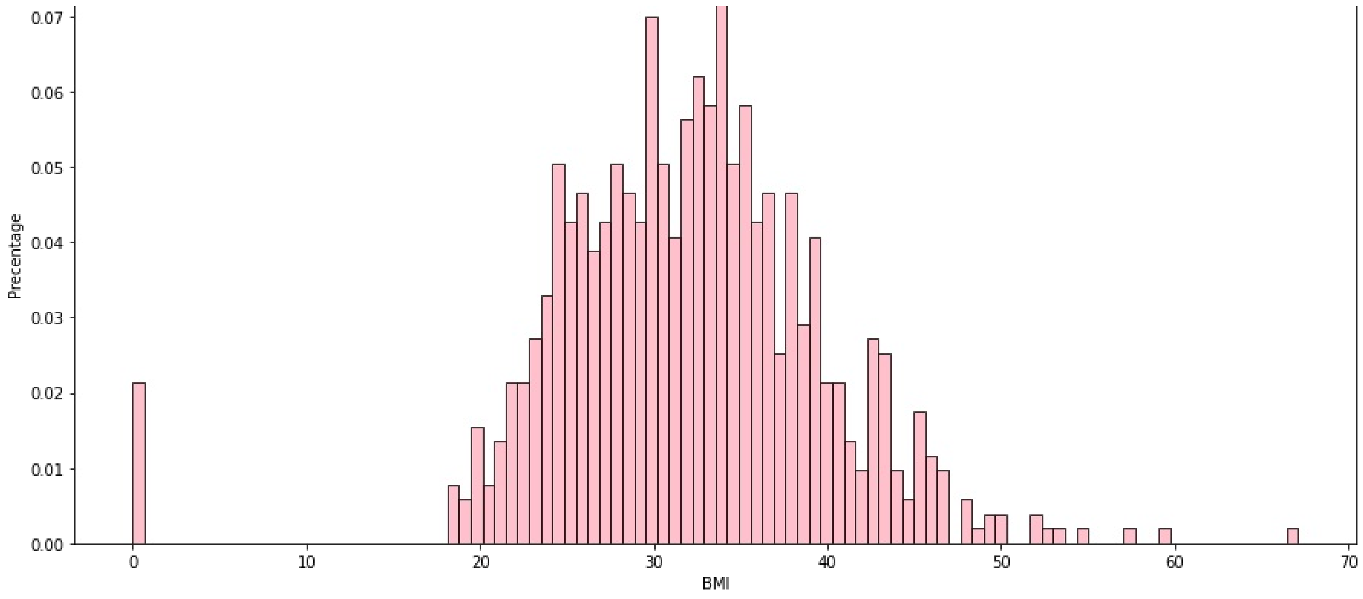


In [13]:
```
#BMI distn

plt.figure(figsize=(15,7))
sns.histplot(data['BMI'], facecolor='pink',bins=100,stat='density')
plt.ylabel('Precentage')
plt.xlabel('BMI')
```

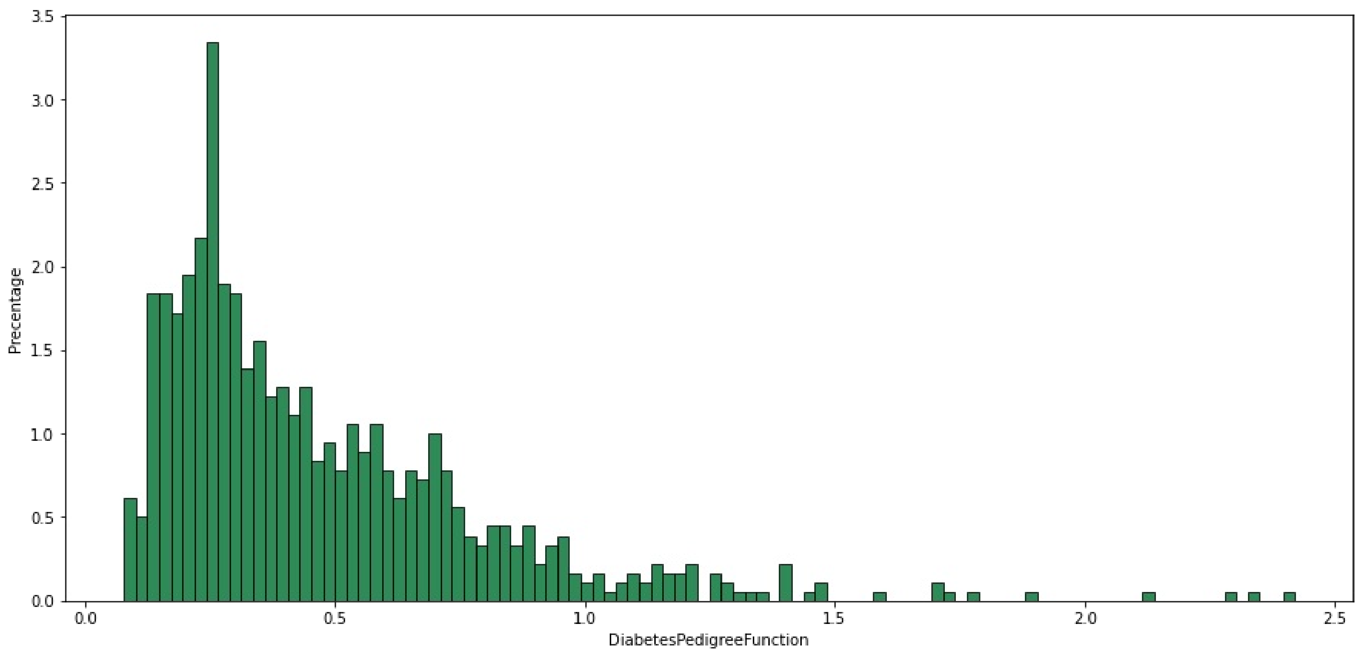
Out[13]:
```
Text(0.5, 0, 'BMI')
```





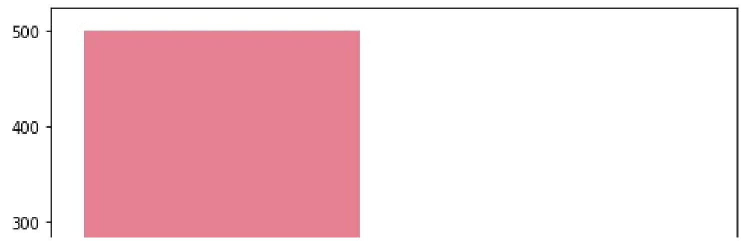
```
In [14]: # Diabetes Pedigree Function EDA
         plt.figure(figsize=(15,7))
         sns.histplot(data['DiabetesPedigreeFunction'], facecolor='seagreen',bins=100,stat='density')
         plt.ylabel('Precentage')
         plt.xlabel('DiabetesPedigreeFunction')
```

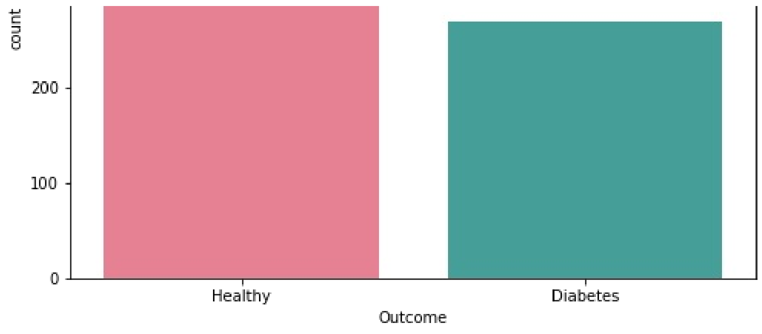
```
Out[14]: Text(0.5, 0, 'DiabetesPedigreeFunction')
```



```
In [15]: from matplotlib.pyplot import figure,show
         figure(figsize=(8,6))
         ax=sns.countplot(x=data['Outcome'],data=data,palette='husl')
         ax.set_xticklabels(['Healthy','Diabetes'])
         healthy,diabetes=data['Outcome'].value_counts().values
         print('Samples of diabetes people:',diabetes)
         print('Samples of healthy people: ',healthy)
```

```
Samples of diabetes people: 268
Samples of healthy people:  500
```

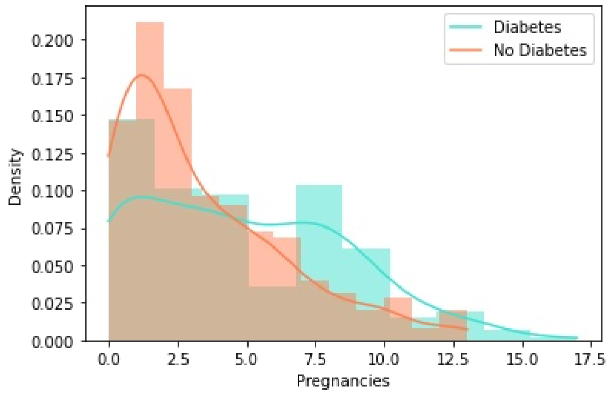




Distribution of other features with respect to Outcome

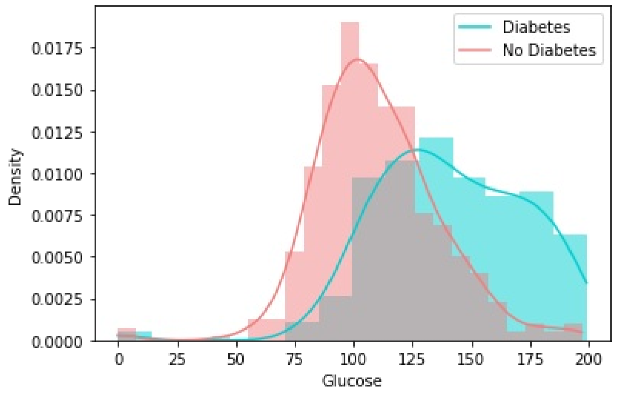
```
In [16]:
plt.figure()
ax=sns.histplot(data['Pregnancies'][data.Outcome==1],color='turquoise',kde=True,stat='density',edgecolor='none')
sns.histplot(data['Pregnancies'][data.Outcome==0],color='coral',kde='True',stat='density',edgecolor='none')
plt.legend(['Diabetes','No Diabetes'])
```

```
Out[16]: <matplotlib.legend.Legend at 0x1de3167f640>
```



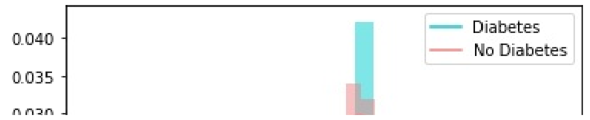
```
In [17]:
plt.figure()
ax=sns.histplot(data['Glucose'][data.Outcome==1],color='darkturquoise',kde=True,stat='density',edgecolor='none')
sns.histplot(data['Glucose'][data.Outcome==0],color='lightcoral',kde='True',stat='density',edgecolor='none')
plt.legend(['Diabetes','No Diabetes'])
```

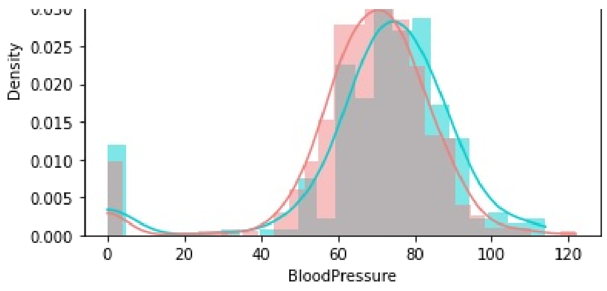
```
Out[17]: <matplotlib.legend.Legend at 0x1de31e430a0>
```



```
In [18]:
plt.figure()
ax=sns.histplot(data['BloodPressure'][data.Outcome==1],color='darkturquoise',kde=True,stat='density',edgecolor='r
sns.histplot(data['BloodPressure'][data.Outcome==0],color='lightcoral',kde='True',stat='density',edgecolor='none'
plt.legend(['Diabetes','No Diabetes'])
```

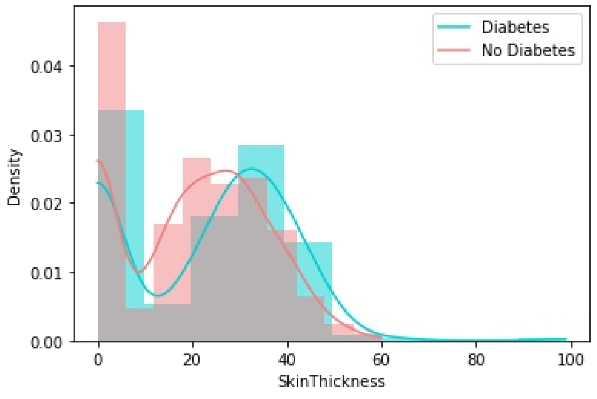
```
Out[18]: <matplotlib.legend.Legend at 0x1de32442100>
```





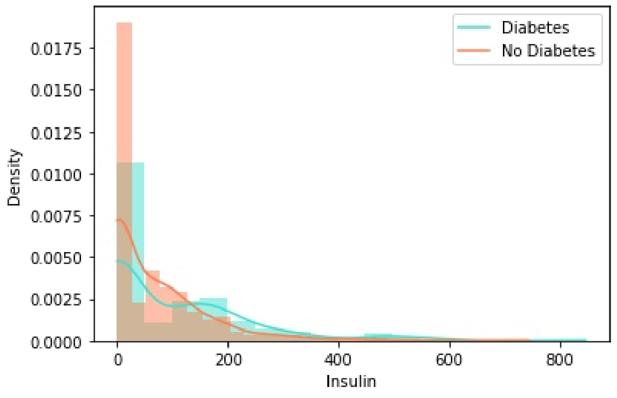
In [19]:
```
plt.figure()
ax=sns.histplot(data['SkinThickness'][data.Outcome==1],color='darkturquoise',kde=True,stat='density',edgecolor='r
sns.histplot(data['SkinThickness'][data.Outcome==0],color='lightcoral',kde='True',stat='density',edgecolor='none'
plt.legend(['Diabetes','No Diabetes'])
```

Out[19]:
```
<matplotlib.legend.Legend at 0x1de324422b0>
```



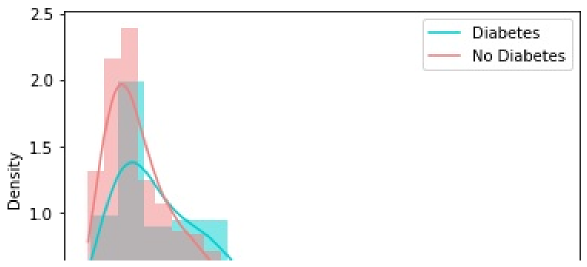
In [20]:
```
plt.figure()
ax=sns.histplot(data['Insulin'][data.Outcome==1],color='turquoise',kde=True,stat='density',edgecolor='none')
sns.histplot(data['Insulin'][data.Outcome==0],color='coral',kde='True',stat='density',edgecolor='none')
plt.legend(['Diabetes','No Diabetes'])
```

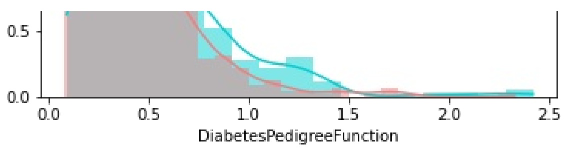
Out[20]:
```
<matplotlib.legend.Legend at 0x1de335c4040>
```



In [21]:
```
plt.figure()
ax=sns.histplot(data['DiabetesPedigreeFunction'][data.Outcome==1],color='darkturquoise',kde=True,stat='density',e
sns.histplot(data['DiabetesPedigreeFunction'][data.Outcome==0],color='lightcoral',kde='True',stat='density',edgec
plt.legend(['Diabetes','No Diabetes'])
```

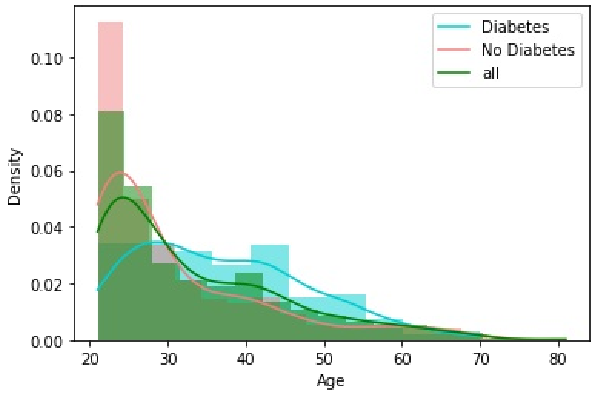
Out[21]:
```
<matplotlib.legend.Legend at 0x1de335a59d0>
```





In [22]:
```
plt.figure()
ax=sns.histplot(data['Age'][data.Outcome==1],color='darkturquoise',kde=True,stat='density',edgecolor='none')
sns.histplot(data['Age'][data.Outcome==0],color='lightcoral',kde='True',stat='density',edgecolor='none')
sns.histplot(data['Age'],color='green',kde=True,stat='density',linewidth=0)
plt.legend(['Diabetes','No Diabetes','all'])
```

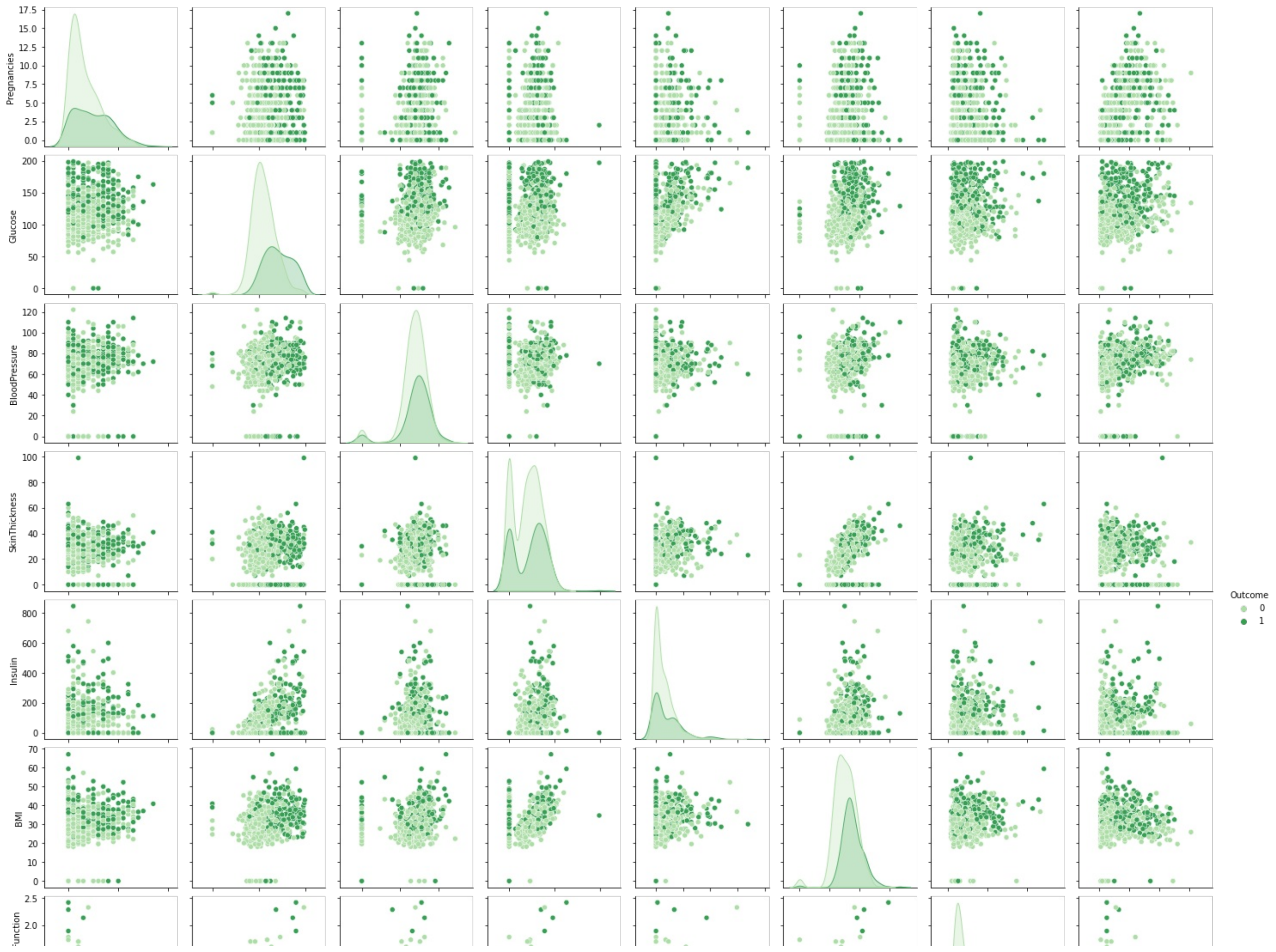
Out[22]:
```
<matplotlib.legend.Legend at 0x1de324aad00>
```

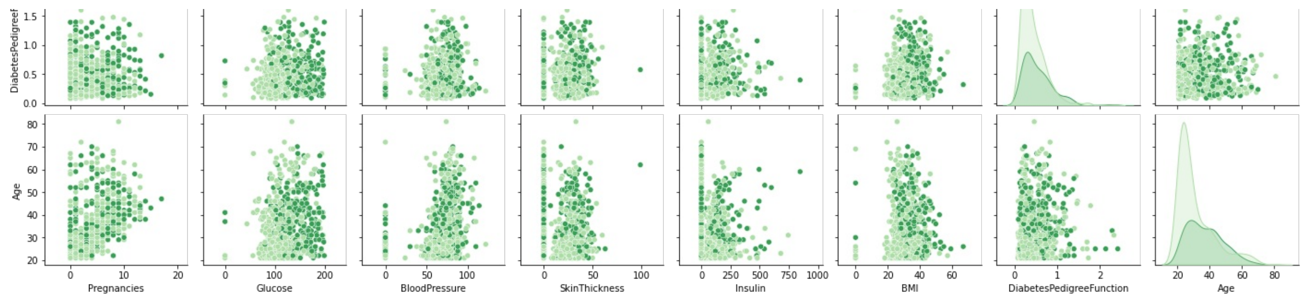


Pairplot

In [23]:
```
sns.pairplot(data,hue='Outcome',palette='Greens')
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x1de31e2c7c0>
```

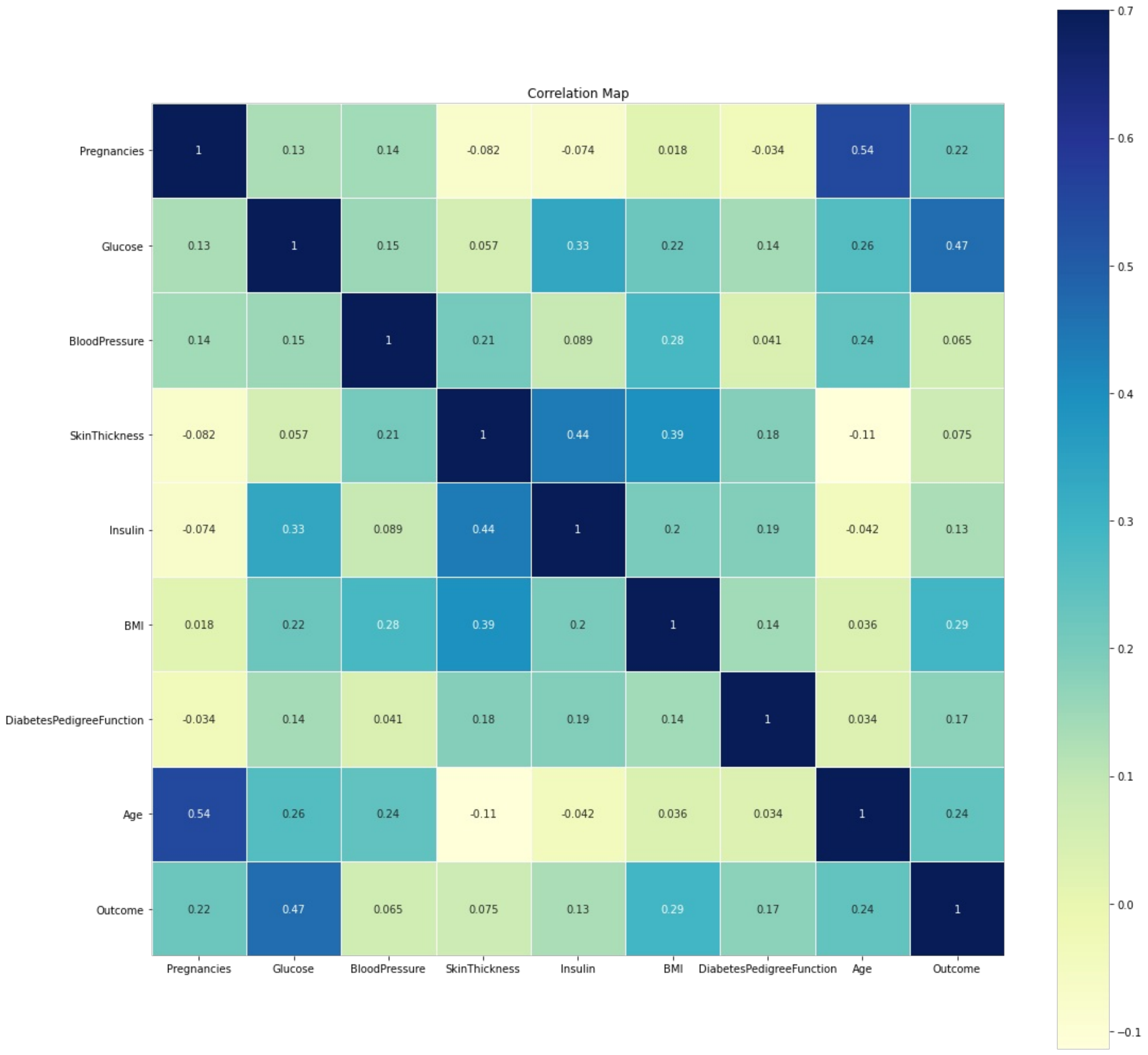




Heatmap Plot Finding Correlation between the columns

In [24]:
```
matrix=data.corr()
f,ax=plt.subplots(figsize=(18,18))
sns.heatmap(matrix,vmax=.7,square=True,cmap='YlGnBu',annot=True,linewidth=.5).set_title('Correlation Map')
```

Out[24]:
```
Text(0.5, 1.0, 'Correlation Map')
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
SkinThickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
Insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

Checking for values which are 0

In [25]:
```
print('Total number of rows in the dataset:',len(data))
print('Total number of rows in the Glucose which are zero:',len(data.loc[data['Glucose']==0]))
print('Total number of rows in the Blood Pressure which are zero:',len(data.loc[data['BloodPressure']==0]))
print('Total number of rows in the Skin Thickness which are zero:',len(data.loc[data['SkinThickness']==0]))
print('Total number of rows in the Insulin which are zero:',len(data.loc[data['Insulin']==0]))
print('Total number of rows in the BMI which are zero:',len(data.loc[data['BMI']==0]))
print('Total number of rows in the DiabetesPedigreeFunction which are zero:',len(data.loc[data['DiabetesPedigreeF
print('Total number of rows in the Age which are zero:',len(data.loc[data['Age']==0]))
```

```
Total number of rows in the dataset: 768
Total number of rows in the Glucose which are zero: 5
Total number of rows in the Blood Pressure which are zero: 35
Total number of rows in the Skin Thickness which are zero: 227
Total number of rows in the Insulin which are zero: 374
Total number of rows in the BMI which are zero: 11
Total number of rows in the DiabetesPedigreeFunction which are zero: 0
Total number of rows in the Age which are zero: 0
```

In [26]:

```
# separating last column i.e outcomes from the dataset and assigning to different variables
data['Outcome'].value_counts()
```

Out[26]:

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

In [27]:

```
data.groupby('Outcome').mean()
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Outcome								
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

Separating data and labels

In [28]:

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

In [29]:

```
print(x)
```

```
     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  
0                       0.627   50  
1                       0.351   31  
2                       0.672   32  
3                       0.167   21  
4                       2.288   33  
..                        ...  ...  
763                     0.171   63  
764                     0.340   27  
765                     0.245   30  
766                     0.349   47  
767                     0.315   23  

[768 rows x 8 columns]
```

In [30]:

```
print(y)
```

```
0      1
1      0
2      1
3      0
```

```
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Data Standardisation

In [31]:
```
scaler=StandardScaler()
scaler.fit(x)
```

Out[31]:
```
StandardScaler()
```

In [32]:
```
standardised_data=scaler.transform(x)
print(standardised_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

In [33]:
```
x=standardised_data
y=data['Outcome']
print(x)
print(y)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Splitting data into train data and test data

In [34]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size= 0.3, stratify=y,random_state=2)
print(x.shape,x_train.shape,x_test.shape)
```

```
(768, 8) (537, 8) (231, 8)
```

Training the model

In [35]:
```
classifier = svm.SVC(kernel='linear')
```

Training the support vector Machine Classifier

In [36]:
```
classifier.fit(x_train,y_train)
```

Out[36]:
```
SVC(kernel='linear')
```

Accuracy score on the training data

In [37]:
```
x_train_prediction = classifier.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
```

In [38]:
```
print('Accuracy score of the training data:',training_data_accuracy)
```

```
Accuracy score of the training data: 0.7821229050279329
```

Making a predictive system

In [39]:
```
input_data=(1,103,30,38,83,43.3,0.183,33) #giving input from dataset and checking whether the prediction is right

input_data_as_numpy_array=np.asarray(input_data) #changing the input data to numpy array

input_data_reshaped=input_data_as_numpy_array.reshape(1,-1) # reshaping the array as we are predicting only for o

std_data=scaler.transform(input_data_reshaped) # standardising the input data
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0]==0):
    print('The person is NOT DIABETIC')
else:
    print('The person is DIABETIC')
```

```
[[-0.84488505 -0.56004775 -2.02166474  1.09545411  0.02778979  1.43512945
  -0.87244072 -0.0204964 ]]
[0]
The person is NOT DIABETIC
```

In [40]:
```
# represents a linear stack of layers
from keras.models import Sequential
# fully connected layer
from keras.layers import Dense

# Model Evaluation
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, recall_score, accura
import pickle
```

In [41]:
```
# Normality Test
from scipy import stats
## retrieving p value from normality test function
PregnanciesPVAL=stats.normaltest(data.Pregnancies).pvalue
GlucosePVAL=stats.normaltest(data.Glucose).pvalue
BloodPressurePVAL=stats.normaltest(data.BloodPressure).pvalue
SkinThicknessPVAL=stats.normaltest(data.SkinThickness).pvalue
InsulinPVAL=stats.normaltest(data.Insulin).pvalue
BMIPVAL=stats.normaltest(data.BMI).pvalue
diaPeFuPVAL=stats.normaltest(data.DiabetesPedigreeFunction).pvalue
AgePVAL=stats.normaltest(data.Age).pvalue
## Printing the values
print("Pregnancies P Value is " + str(PregnanciesPVAL))
print("Glucose P Value is " + str(GlucosePVAL))
print("BloodPressure P Value is " + str(BloodPressurePVAL))
print("Skin Thickness P Value is " + str(SkinThicknessPVAL))
print("Insulin P Value is " + str(InsulinPVAL))
print("BMI P Value is " + str(BMIPVAL))
print("dfbetes Pedigree Function P Value is " + str(diaPeFuPVAL))
print("Age P Value is " + str(AgePVAL))
```

```
Pregnancies P Value is 3.9142916387836715e-18
Glucose P Value is 0.0020446506991363502
BloodPressure P Value is 3.780127075543807e-67
Skin Thickness P Value is 0.00017124157678801767
Insulin P Value is 6.895342740914799e-85
BMI P Value is 1.9696869549260572e-19
dfbetes Pedigree Function P Value is 1.2987697487612696e-70
Age P Value is 9.308980038236578e-27
```

Conclusion

None of the variables are normal. (P>0.05)

The variables "Pregnancies," "Glucose," "Insulin," "BMI," "Diabetes Pedigree Function," and "Age" show strong evidence of association with the outcome variable based on their small p-values. The variables "Blood Pressure" and "Skin Thickness" also show some evidence of association, although the evidence may not be as strong.

In [42]:

```
import plotly.graph_objs as go
textd = ['non-diabetic' if cl==0 else 'diabetic' for cl in data['Outcome']]

fig = go.Figure(data=go.Splom(
                  dimensions=[dict(label='Pregnancies', values=data['Pregnancies']),
                              dict(label='Glucose', values=data['Glucose']),
                              dict(label='BloodPressure', values=data['BloodPressure']),
                              dict(label='SkinThickness', values=data['SkinThickness']),
                              dict(label='Insulin', values=data['Insulin']),
                              dict(label='BMI', values=data['BMI']),
                              dict(label='DiabPedigreeFun', values=data['DiabetesPedigreeFunction']),
                              dict(label='Age', values=data['Age'])],
                  showupperhalf=False,# remove plots on diagonal
                  marker=dict(color=data['Outcome'],
                              size=5,
                              colorscale='Viridis',
                              line=dict(width=0.5,
                                        color='rgb(230,230,230)')),
                  text=textd))
                  #diagonal=dict(visible=False)))

title = "Scatterplot Matrix for Diabetes Dataset"
fig.update_layout(title=title,
                  template = 'plotly_white',
                  dragmode='select',
                  width=1000,
                  height=1000,
                  hovermode='closest')

fig.show()
```

WebGL is not supported by your browser
- visit https://get.webgl.org for more

info

Note: In the above boxplots yellow points refers Diabetic and dark blue points refers Non-diabetic

Conclusion

Only BMI & SkinThickness and Pregnancies & Age seem to have positive linear relationships. Another likely suspect is Glucose and Insulin. There are no non-linear relationships.

Thank You

In [ ]: