Importing Libraries

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

sns.set()

from mlxtend.plotting import plot_decision_regions
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Importing Dataset

Data Characteristics

In [2]:
```
df = pd.read_csv('diabetes_data.csv')
df = df.drop(columns=['Unnamed: 0'])
```

In [3]:
```
# Exploring Dataset Columns
df.columns
```

Out[3]:
```
Index(['Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose', 'HbA1c',
       'Random_Blood_Glucose', 'BMI', 'Waist_Circumference',
       'Triglyceride_Levels', 'Blood_Pressure_Systolic',
       'Blood_Pressure_Diastolic', 'LDL_Cholesterol', 'HDL_Cholesterol',
       'CRP_Levels', 'Insulin_Levels', 'HOMA_IR', 'OGTT', 'Creatinine_Levels',
       'eGFR', 'Microalbuminuria', 'Uric_Acid_Levels', 'Fructosamine_Levels',
       'ALT', 'AST', 'C_Peptide', 'Proinsulin_Levels',
       'Family_History_of_Diabetes', 'Gestational_Diabetes', 'PCOS',
       'Hypertension', 'Physical_Activity', 'Smoking', 'Alcohol_Consumption',
       'Obesity', 'Diet', 'Sleep_Apnea', 'Diabetes_Status'],
      dtype='object')
```

In [4]:
```
# Dataset Information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 35 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Fasting_Blood_Glucose        3000 non-null   float64
 1   Postprandial_Blood_Glucose   3000 non-null   float64
 2   HbA1c                        3000 non-null   float64
 3   Random_Blood_Glucose         3000 non-null   float64
 4   BMI                          3000 non-null   float64
 5   Waist_Circumference          3000 non-null   float64
 6   Triglyceride_Levels          3000 non-null   float64
 7   Blood_Pressure_Systolic      3000 non-null   float64
 8   Blood_Pressure_Diastolic     3000 non-null   float64
 9   LDL_Cholesterol              3000 non-null   float64
 10  HDL_Cholesterol              3000 non-null   float64
 11  CRP_Levels                   3000 non-null   float64
 12  Insulin_Levels               3000 non-null   float64
 13  HOMA_IR                      3000 non-null   float64
 14  OGTT                         3000 non-null   float64
 15  Creatinine_Levels            3000 non-null   float64
 16  eGFR                         3000 non-null   float64
 17  Microalbuminuria             3000 non-null   float64
 18  Uric_Acid_Levels             3000 non-null   float64
 19  Fructosamine_Levels          3000 non-null   float64
 20  ALT                          3000 non-null   float64
 21  AST                          3000 non-null   float64
 22  C_Peptide                    3000 non-null   float64
 23  Proinsulin_Levels            3000 non-null   float64
 24  Family_History_of_Diabetes   3000 non-null   object 
 25  Gestational_Diabetes         3000 non-null   object 
 26  PCOS                         3000 non-null   object 
 27  Hypertension                 3000 non-null   object 
 28  Physical_Activity            3000 non-null   object 
 29  Smoking                      3000 non-null   object 
 30  Alcohol_Consumption          3000 non-null   object 
 31  Obesity                      3000 non-null   object 
 32  Diet                         3000 non-null   object 
 33  Sleep_Apnea                  3000 non-null   object 
 34  Diabetes_Status              3000 non-null   object 
dtypes: float64(24), object(11)
memory usage: 820.4+ KB
```

Dataset Preprocessing

To understand the statistics of the dataset:

Checking for Null Values

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Fasting_Blood_Glucose          0
Postprandial_Blood_Glucose     0
HbA1c                          0
Random_Blood_Glucose           0
BMI                            0
Waist_Circumference            0
Triglyceride_Levels            0
Blood_Pressure_Systolic        0
Blood_Pressure_Diastolic       0
LDL_Cholesterol                0
HDL_Cholesterol                0
CRP_Levels                     0
Insulin_Levels                 0
HOMA_IR                        0
OGTT                           0
Creatinine_Levels              0
eGFR                           0
Microalbuminuria               0
Uric_Acid_Levels               0
Fructosamine_Levels            0
ALT                            0
AST                            0
C_Peptide                      0
Proinsulin_Levels              0
Family_History_of_Diabetes     0
Gestational_Diabetes           0
PCOS                           0
Hypertension                   0
Physical_Activity              0
Smoking                        0
Alcohol_Consumption            0
Obesity                        0
Diet                           0
Sleep_Apnea                    0
Diabetes_Status                0
dtype: int64
```

Checking For The duplicate Values

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
# df.to_csv('diabetes_data.csv')
```

Standardise Binary Variables

Ensured binary variables are consistently encoded (e.g., 'Yes' as 1, 'No' as 0, and Positive as 1, Negative as0).

In [8]:
```
df['Diabetes_Status'] = df['Diabetes_Status'].replace({'Positive':1,'Negative':0})

df = df.replace({'Yes':1,'No':0})
```

Splitting the Dataset

In [9]:
```
X = df.drop('Diabetes_Status',axis=1)
y = df['Diabetes_Status']

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=42)
```

Scaling the Data

Applying Standard Scaling

In [10]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Exploratory Data Analysis (EDA)

Summary Statistics

In [11]:
```
df.describe().T
```

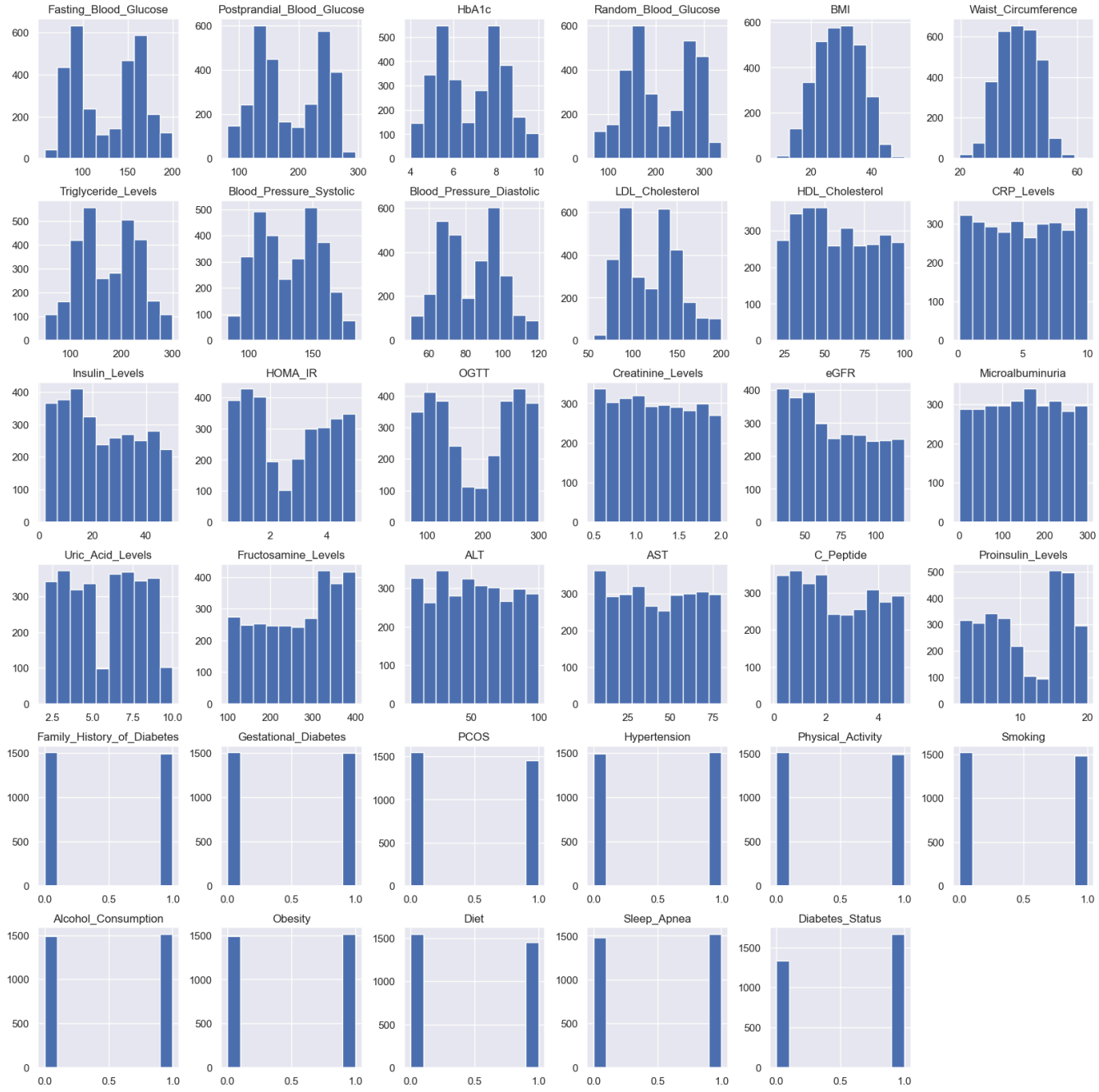
Out[11]:

	count	mean	std	min	25%	50%	75%	
Fasting_Blood_Glucose	3000.0	127.882942	37.131397	57.630063	91.333390	134.393435	160.823524	199
Postprandial_Blood_Glucose	3000.0	185.252812	55.952564	80.547105	137.702202	171.799770	242.320433	295
HbA1c	3000.0	6.830455	1.507592	4.000070	5.465759	6.959233	8.095574	9
Random_Blood_Glucose	3000.0	207.759310	66.975314	70.150280	152.618601	194.424066	273.470407	336
BMI	3000.0	29.174754	7.183878	8.673494	23.785948	29.300126	34.668594	50
Waist_Circumference	3000.0	39.851831	6.592802	19.619019	34.774728	39.726221	44.998099	63
Triglyceride_Levels	3000.0	174.460457	57.796175	50.357086	127.692607	173.257466	222.733259	299
Blood_Pressure_Systolic	3000.0	131.248308	23.597147	83.561186	110.320283	131.038994	150.941414	183
Blood_Pressure_Diastolic	3000.0	83.034597	15.846254	50.017005	69.619610	83.224700	95.670146	119
LDL_Cholesterol	3000.0	121.381494	31.835540	55.800940	92.228701	125.286285	143.608020	199
HDL_Cholesterol	3000.0	58.011141	22.783630	20.000000	38.841806	56.003185	78.000000	99
CRP_Levels	3000.0	5.055953	2.920785	0.101562	2.515998	4.995072	7.591052	9
Insulin_Levels	3000.0	23.591630	13.984759	2.000000	11.937146	22.000000	36.000000	49
HOMA_IR	3000.0	2.667320	1.425389	0.502343	1.334220	2.670937	3.990770	4
OGTT	3000.0	185.621117	74.349777	70.000000	115.000000	185.221876	257.000000	299
Creatinine_Levels	3000.0	1.229341	0.432980	0.500465	0.856201	1.213373	1.607590	1
eGFR	3000.0	69.951377	26.413320	30.000000	47.000000	67.000000	92.620456	119
Microalbuminuria	3000.0	150.117338	85.393535	0.000000	78.488599	151.122273	223.944261	299
Uric_Acid_Levels	3000.0	5.683695	2.314232	2.000000	3.921496	6.000000	7.878333	9
Fructosamine_Levels	3000.0	265.399634	89.422951	100.000000	186.360327	278.001549	344.000000	399
ALT	3000.0	51.493356	27.344695	5.000000	28.000000	51.000000	75.000000	99
AST	3000.0	41.648658	22.160046	5.000000	22.000000	41.000000	61.000000	79
C_Peptide	3000.0	2.435448	1.446943	0.102499	1.156524	2.298546	3.741384	4
Proinsulin_Levels	3000.0	10.843602	6.043202	1.000000	5.000000	10.412237	17.000000	19
Family_History_of_Diabetes	3000.0	0.498000	0.500079	0.000000	0.000000	0.000000	1.000000	1
Gestational_Diabetes	3000.0	0.498667	0.500082	0.000000	0.000000	0.000000	1.000000	1
PCOS	3000.0	0.484333	0.499838	0.000000	0.000000	0.000000	1.000000	1
Hypertension	3000.0	0.501667	0.500081	0.000000	0.000000	1.000000	1.000000	1
Physical_Activity	3000.0	0.497000	0.500074	0.000000	0.000000	0.000000	1.000000	1
Smoking	3000.0	0.493333	0.500039	0.000000	0.000000	0.000000	1.000000	1
Alcohol_Consumption	3000.0	0.503333	0.500072	0.000000	0.000000	1.000000	1.000000	1
Obesity	3000.0	0.504000	0.500067	0.000000	0.000000	1.000000	1.000000	1
Diet	3000.0	0.484667	0.499848	0.000000	0.000000	0.000000	1.000000	1
Sleep_Apnea	3000.0	0.505667	0.500051	0.000000	0.000000	1.000000	1.000000	1
Diabetes_Status	3000.0	0.554667	0.497085	0.000000	0.000000	1.000000	1.000000	1

Data Visualization

Histogram

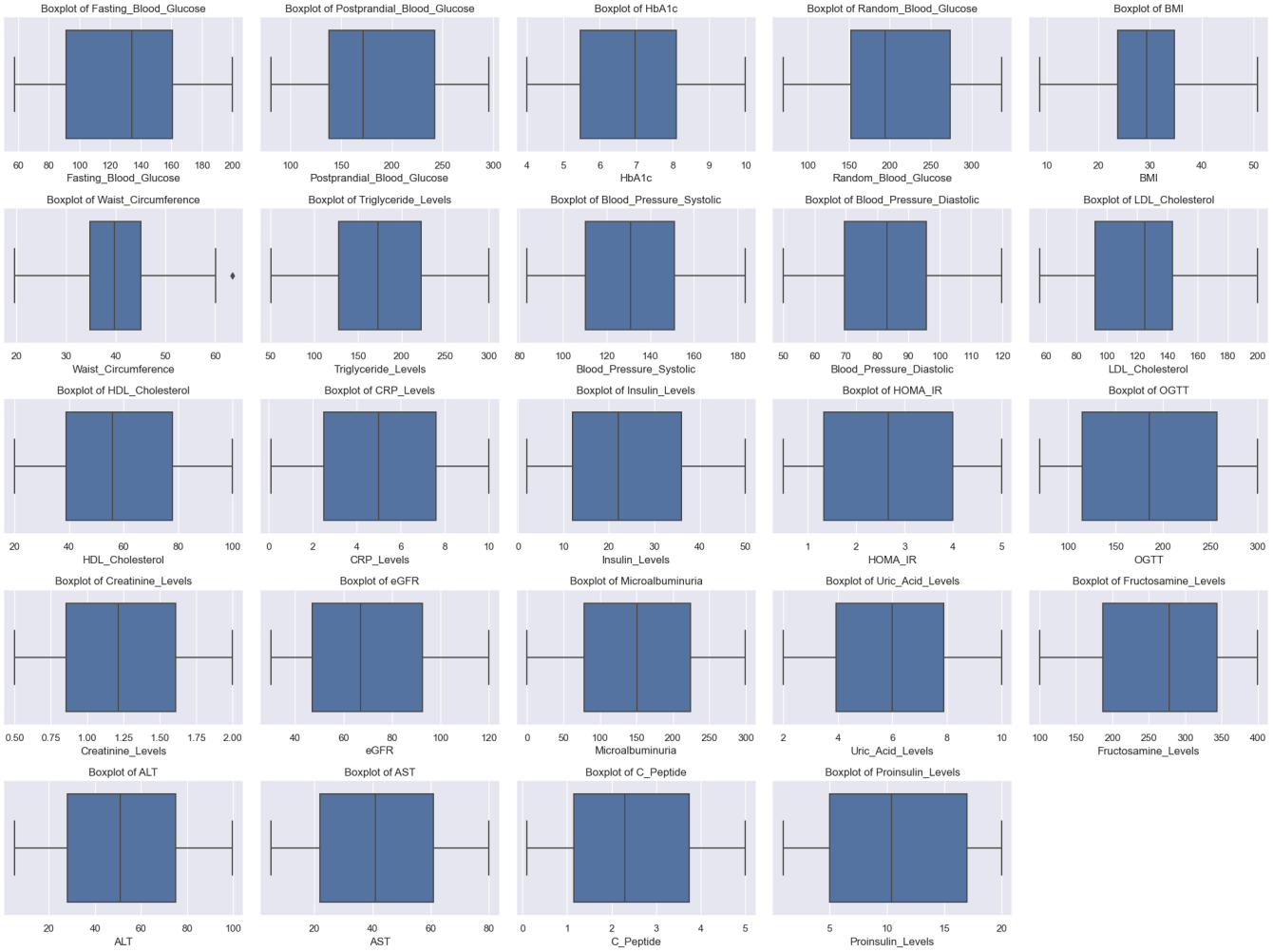
In [12]:
```
df.hist(figsize=(20,20))
plt.show()
```



Boxplot

In [13]:
```
numeric_columns = [
    'Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose', 'HbA1c', 'Random_Blood_Glucose', '
    'Waist_Circumference', 'Triglyceride_Levels', 'Blood_Pressure_Systolic', 'Blood_Pressure_
    'LDL_Cholesterol', 'HDL_Cholesterol', 'CRP_Levels', 'Insulin_Levels', 'HOMA_IR', 'OGTT',
    'Creatinine_Levels', 'eGFR', 'Microalbuminuria', 'Uric_Acid_Levels', 'Fructosamine_Levels
    'ALT', 'AST', 'C_Peptide', 'Proinsulin_Levels'
]
plt.figure(figsize=(20,15))
for i, col in enumerate(numeric_columns):
    plt.subplot(5, 5, i+1)
    sns.boxplot(data=df, x=col, hue='Diabetes_Status')
    plt.title(f'Boxplot of {col}')
```

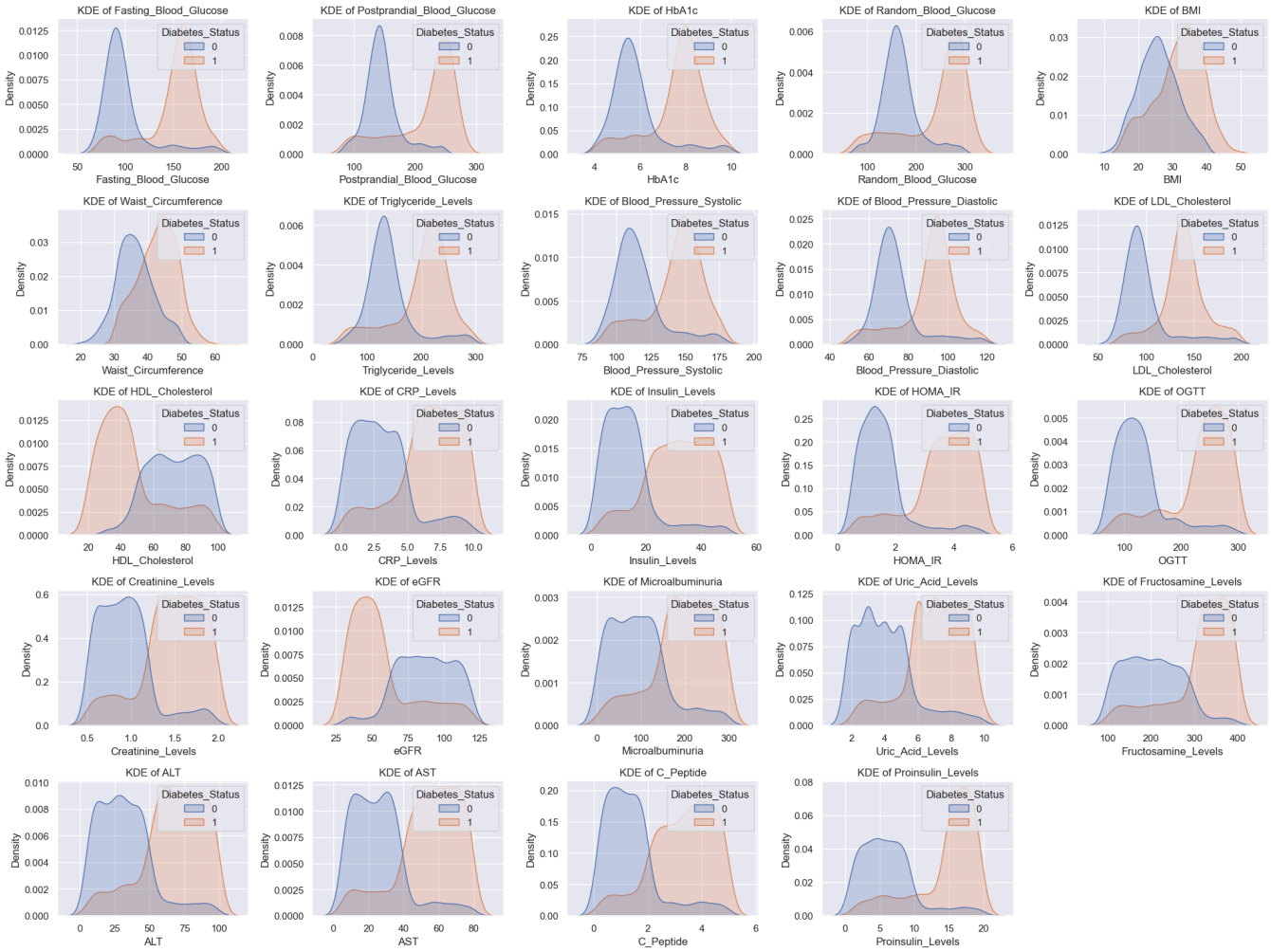
```
plt.tight_layout()
plt.show()
```



kdeplot

Distribution of the Random Variables

In [14]:
```
plt.figure(figsize=(20, 15))
numeric_columns = [
    'Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose', 'HbA1c', 'Random_Blood_Glucose', '
    'Waist_Circumference', 'Triglyceride_Levels', 'Blood_Pressure_Systolic', 'Blood_Pressure_
    'LDL_Cholesterol', 'HDL_Cholesterol', 'CRP_Levels', 'Insulin_Levels', 'HOMA_IR', 'OGTT',
    'Creatinine_Levels', 'eGFR', 'Microalbuminuria', 'Uric_Acid_Levels', 'Fructosamine_Levels
    'ALT', 'AST', 'C_Peptide', 'Proinsulin_Levels'
]

for i, col in enumerate(numeric_columns):
    plt.subplot(5, 5, i+1)
    sns.kdeplot(data=df, x=col, hue='Diabetes_Status', fill=True)
    plt.title(f'KDE of {col}')
plt.tight_layout()
plt.show()
```



In [15]:

```
numeric_columns = [
    'Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose', 'HbA1c', 'Random_Blood_Glucose', '
    'Waist_Circumference', 'Triglyceride_Levels', 'Blood_Pressure_Systolic', 'Blood_Pressure_
    'LDL_Cholesterol', 'HDL_Cholesterol', 'CRP_Levels', 'Insulin_Levels', 'HOMA_IR', 'OGTT',
    'Creatinine_Levels', 'eGFR', 'Microalbuminuria', 'Uric_Acid_Levels', 'Fructosamine_Levels
    'ALT', 'AST', 'C_Peptide', 'Proinsulin_Levels'
]

lst = []

for col1 in numeric_columns:
    for col2 in numeric_columns:
        if col1 != col2:
            sorted_pair = sorted([col1, col2])
            if sorted_pair not in lst:
                lst.append(sorted_pair)
lst
```

Out[15]:
```
[['Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'HbA1c'],
 ['Fasting_Blood_Glucose', 'Random_Blood_Glucose'],
 ['BMI', 'Fasting_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'Waist_Circumference'],
 ['Fasting_Blood_Glucose', 'Triglyceride_Levels'],
 ['Blood_Pressure_Systolic', 'Fasting_Blood_Glucose'],
 ['Blood_Pressure_Diastolic', 'Fasting_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'LDL_Cholesterol'],
 ['Fasting_Blood_Glucose', 'HDL_Cholesterol'],
 ['CRP_Levels', 'Fasting_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'Insulin_Levels'],
 ['Fasting_Blood_Glucose', 'HOMA_IR'],
 ['Fasting_Blood_Glucose', 'OGTT'],
 ['Creatinine_Levels', 'Fasting_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'eGFR'],
 ['Fasting_Blood_Glucose', 'Microalbuminuria'],
 ['Fasting_Blood_Glucose', 'Uric_Acid_Levels'],
 ['Fasting_Blood_Glucose', 'Fructosamine_Levels'],
 ['ALT', 'Fasting_Blood_Glucose'],
 ['AST', 'Fasting_Blood_Glucose'],
 ['C_Peptide', 'Fasting_Blood_Glucose'],
 ['Fasting_Blood_Glucose', 'Proinsulin_Levels'],
 ['HbA1c', 'Postprandial_Blood_Glucose'],
 ['Postprandial_Blood_Glucose', 'Random_Blood_Glucose'],
 ['BMI', 'Postprandial_Blood_Glucose'],
 ['Postprandial_Blood_Glucose', 'Waist_Circumference'],
 ['Postprandial_Blood_Glucose', 'Triglyceride_Levels'],
 ['Blood_Pressure_Systolic', 'Postprandial_Blood_Glucose'],
 ['Blood_Pressure_Diastolic', 'Postprandial_Blood_Glucose'],
 ['LDL_Cholesterol', 'Postprandial_Blood_Glucose'],
 ['HDL_Cholesterol', 'Postprandial_Blood_Glucose'],
 ['CRP_Levels', 'Postprandial_Blood_Glucose'],
 ['Insulin_Levels', 'Postprandial_Blood_Glucose'],
 ['HOMA_IR', 'Postprandial_Blood_Glucose'],
 ['OGTT', 'Postprandial_Blood_Glucose'],
 ['Creatinine_Levels', 'Postprandial_Blood_Glucose'],
 ['Postprandial_Blood_Glucose', 'eGFR'],
 ['Microalbuminuria', 'Postprandial_Blood_Glucose'],
 ['Postprandial_Blood_Glucose', 'Uric_Acid_Levels'],
 ['Fructosamine_Levels', 'Postprandial_Blood_Glucose'],
 ['ALT', 'Postprandial_Blood_Glucose'],
 ['AST', 'Postprandial_Blood_Glucose'],
 ['C_Peptide', 'Postprandial_Blood_Glucose'],
 ['Postprandial_Blood_Glucose', 'Proinsulin_Levels'],
 ['HbA1c', 'Random_Blood_Glucose'],
 ['BMI', 'HbA1c'],
 ['HbA1c', 'Waist_Circumference'],
 ['HbA1c', 'Triglyceride_Levels'],
 ['Blood_Pressure_Systolic', 'HbA1c'],
 ['Blood_Pressure_Diastolic', 'HbA1c'],
 ['HbA1c', 'LDL_Cholesterol'],
 ['HDL_Cholesterol', 'HbA1c'],
 ['CRP_Levels', 'HbA1c'],
 ['HbA1c', 'Insulin_Levels'],
 ['HOMA_IR', 'HbA1c'],
 ['HbA1c', 'OGTT'],
 ['Creatinine_Levels', 'HbA1c'],
 ['HbA1c', 'eGFR'],
 ['HbA1c', 'Microalbuminuria'],
 ['HbA1c', 'Uric_Acid_Levels'],
 ['Fructosamine_Levels', 'HbA1c'],
 ['ALT', 'HbA1c'],
 ['AST', 'HbA1c'],
 ['C_Peptide', 'HbA1c'],
 ['HbA1c', 'Proinsulin_Levels'],
```

```
['BMI', 'Random_Blood_Glucose'],
['Random_Blood_Glucose', 'Waist_Circumference'],
['Random_Blood_Glucose', 'Triglyceride_Levels'],
['Blood_Pressure_Systolic', 'Random_Blood_Glucose'],
['Blood_Pressure_Diastolic', 'Random_Blood_Glucose'],
['LDL_Cholesterol', 'Random_Blood_Glucose'],
['HDL_Cholesterol', 'Random_Blood_Glucose'],
['CRP_Levels', 'Random_Blood_Glucose'],
['Insulin_Levels', 'Random_Blood_Glucose'],
['HOMA_IR', 'Random_Blood_Glucose'],
['OGTT', 'Random_Blood_Glucose'],
['Creatinine_Levels', 'Random_Blood_Glucose'],
['Random_Blood_Glucose', 'eGFR'],
['Microalbuminuria', 'Random_Blood_Glucose'],
['Random_Blood_Glucose', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'Random_Blood_Glucose'],
['ALT', 'Random_Blood_Glucose'],
['AST', 'Random_Blood_Glucose'],
['C_Peptide', 'Random_Blood_Glucose'],
['Proinsulin_Levels', 'Random_Blood_Glucose'],
['BMI', 'Waist_Circumference'],
['BMI', 'Triglyceride_Levels'],
['BMI', 'Blood_Pressure_Systolic'],
['BMI', 'Blood_Pressure_Diastolic'],
['BMI', 'LDL_Cholesterol'],
['BMI', 'HDL_Cholesterol'],
['BMI', 'CRP_Levels'],
['BMI', 'Insulin_Levels'],
['BMI', 'HOMA_IR'],
['BMI', 'OGTT'],
['BMI', 'Creatinine_Levels'],
['BMI', 'eGFR'],
['BMI', 'Microalbuminuria'],
['BMI', 'Uric_Acid_Levels'],
['BMI', 'Fructosamine_Levels'],
['ALT', 'BMI'],
['AST', 'BMI'],
['BMI', 'C_Peptide'],
['BMI', 'Proinsulin_Levels'],
['Triglyceride_Levels', 'Waist_Circumference'],
['Blood_Pressure_Systolic', 'Waist_Circumference'],
['Blood_Pressure_Diastolic', 'Waist_Circumference'],
['LDL_Cholesterol', 'Waist_Circumference'],
['HDL_Cholesterol', 'Waist_Circumference'],
['CRP_Levels', 'Waist_Circumference'],
['Insulin_Levels', 'Waist_Circumference'],
['HOMA_IR', 'Waist_Circumference'],
['OGTT', 'Waist_Circumference'],
['Creatinine_Levels', 'Waist_Circumference'],
['Waist_Circumference', 'eGFR'],
['Microalbuminuria', 'Waist_Circumference'],
['Uric_Acid_Levels', 'Waist_Circumference'],
['Fructosamine_Levels', 'Waist_Circumference'],
['ALT', 'Waist_Circumference'],
['AST', 'Waist_Circumference'],
['C_Peptide', 'Waist_Circumference'],
['Proinsulin_Levels', 'Waist_Circumference'],
['Blood_Pressure_Systolic', 'Triglyceride_Levels'],
['Blood_Pressure_Diastolic', 'Triglyceride_Levels'],
['LDL_Cholesterol', 'Triglyceride_Levels'],
['HDL_Cholesterol', 'Triglyceride_Levels'],
['CRP_Levels', 'Triglyceride_Levels'],
['Insulin_Levels', 'Triglyceride_Levels'],
['HOMA_IR', 'Triglyceride_Levels'],
['OGTT', 'Triglyceride_Levels'],
['Creatinine_Levels', 'Triglyceride_Levels'],
```

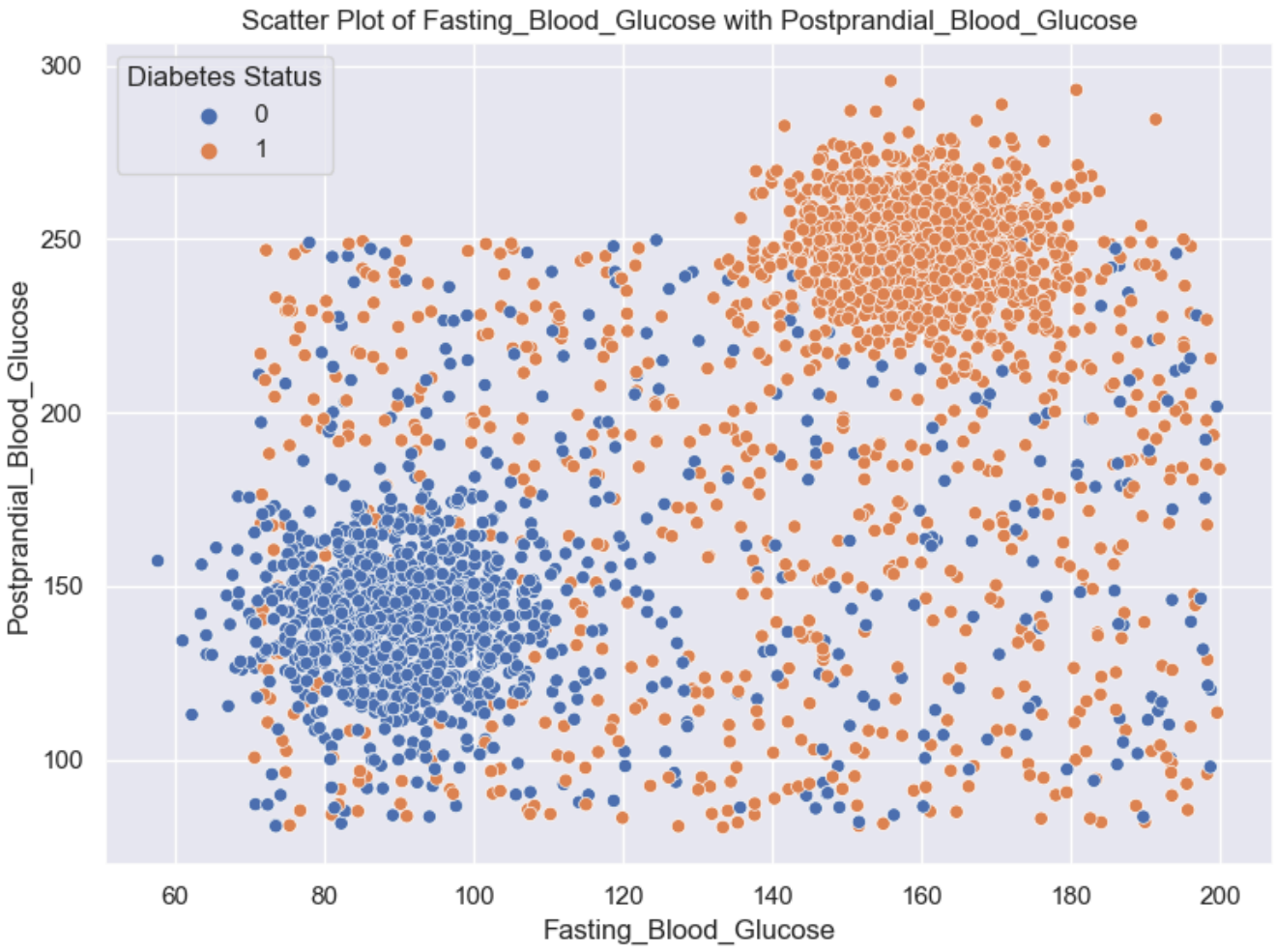
```
['Triglyceride_Levels', 'eGFR'],
['Microalbuminuria', 'Triglyceride_Levels'],
['Triglyceride_Levels', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'Triglyceride_Levels'],
['ALT', 'Triglyceride_Levels'],
['AST', 'Triglyceride_Levels'],
['C_Peptide', 'Triglyceride_Levels'],
['Proinsulin_Levels', 'Triglyceride_Levels'],
['Blood_Pressure_Diastolic', 'Blood_Pressure_Systolic'],
['Blood_Pressure_Systolic', 'LDL_Cholesterol'],
['Blood_Pressure_Systolic', 'HDL_Cholesterol'],
['Blood_Pressure_Systolic', 'CRP_Levels'],
['Blood_Pressure_Systolic', 'Insulin_Levels'],
['Blood_Pressure_Systolic', 'HOMA_IR'],
['Blood_Pressure_Systolic', 'OGTT'],
['Blood_Pressure_Systolic', 'Creatinine_Levels'],
['Blood_Pressure_Systolic', 'eGFR'],
['Blood_Pressure_Systolic', 'Microalbuminuria'],
['Blood_Pressure_Systolic', 'Uric_Acid_Levels'],
['Blood_Pressure_Systolic', 'Fructosamine_Levels'],
['ALT', 'Blood_Pressure_Systolic'],
['AST', 'Blood_Pressure_Systolic'],
['Blood_Pressure_Systolic', 'C_Peptide'],
['Blood_Pressure_Systolic', 'Proinsulin_Levels'],
['Blood_Pressure_Diastolic', 'LDL_Cholesterol'],
['Blood_Pressure_Diastolic', 'HDL_Cholesterol'],
['Blood_Pressure_Diastolic', 'CRP_Levels'],
['Blood_Pressure_Diastolic', 'Insulin_Levels'],
['Blood_Pressure_Diastolic', 'HOMA_IR'],
['Blood_Pressure_Diastolic', 'OGTT'],
['Blood_Pressure_Diastolic', 'Creatinine_Levels'],
['Blood_Pressure_Diastolic', 'eGFR'],
['Blood_Pressure_Diastolic', 'Microalbuminuria'],
['Blood_Pressure_Diastolic', 'Uric_Acid_Levels'],
['Blood_Pressure_Diastolic', 'Fructosamine_Levels'],
['ALT', 'Blood_Pressure_Diastolic'],
['AST', 'Blood_Pressure_Diastolic'],
['Blood_Pressure_Diastolic', 'C_Peptide'],
['Blood_Pressure_Diastolic', 'Proinsulin_Levels'],
['HDL_Cholesterol', 'LDL_Cholesterol'],
['CRP_Levels', 'LDL_Cholesterol'],
['Insulin_Levels', 'LDL_Cholesterol'],
['HOMA_IR', 'LDL_Cholesterol'],
['LDL_Cholesterol', 'OGTT'],
['Creatinine_Levels', 'LDL_Cholesterol'],
['LDL_Cholesterol', 'eGFR'],
['LDL_Cholesterol', 'Microalbuminuria'],
['LDL_Cholesterol', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'LDL_Cholesterol'],
['ALT', 'LDL_Cholesterol'],
['AST', 'LDL_Cholesterol'],
['C_Peptide', 'LDL_Cholesterol'],
['LDL_Cholesterol', 'Proinsulin_Levels'],
['CRP_Levels', 'HDL_Cholesterol'],
['HDL_Cholesterol', 'Insulin_Levels'],
['HDL_Cholesterol', 'HOMA_IR'],
['HDL_Cholesterol', 'OGTT'],
['Creatinine_Levels', 'HDL_Cholesterol'],
['HDL_Cholesterol', 'eGFR'],
['HDL_Cholesterol', 'Microalbuminuria'],
['HDL_Cholesterol', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'HDL_Cholesterol'],
['ALT', 'HDL_Cholesterol'],
['AST', 'HDL_Cholesterol'],
['C_Peptide', 'HDL_Cholesterol'],
['HDL_Cholesterol', 'Proinsulin_Levels'],
```

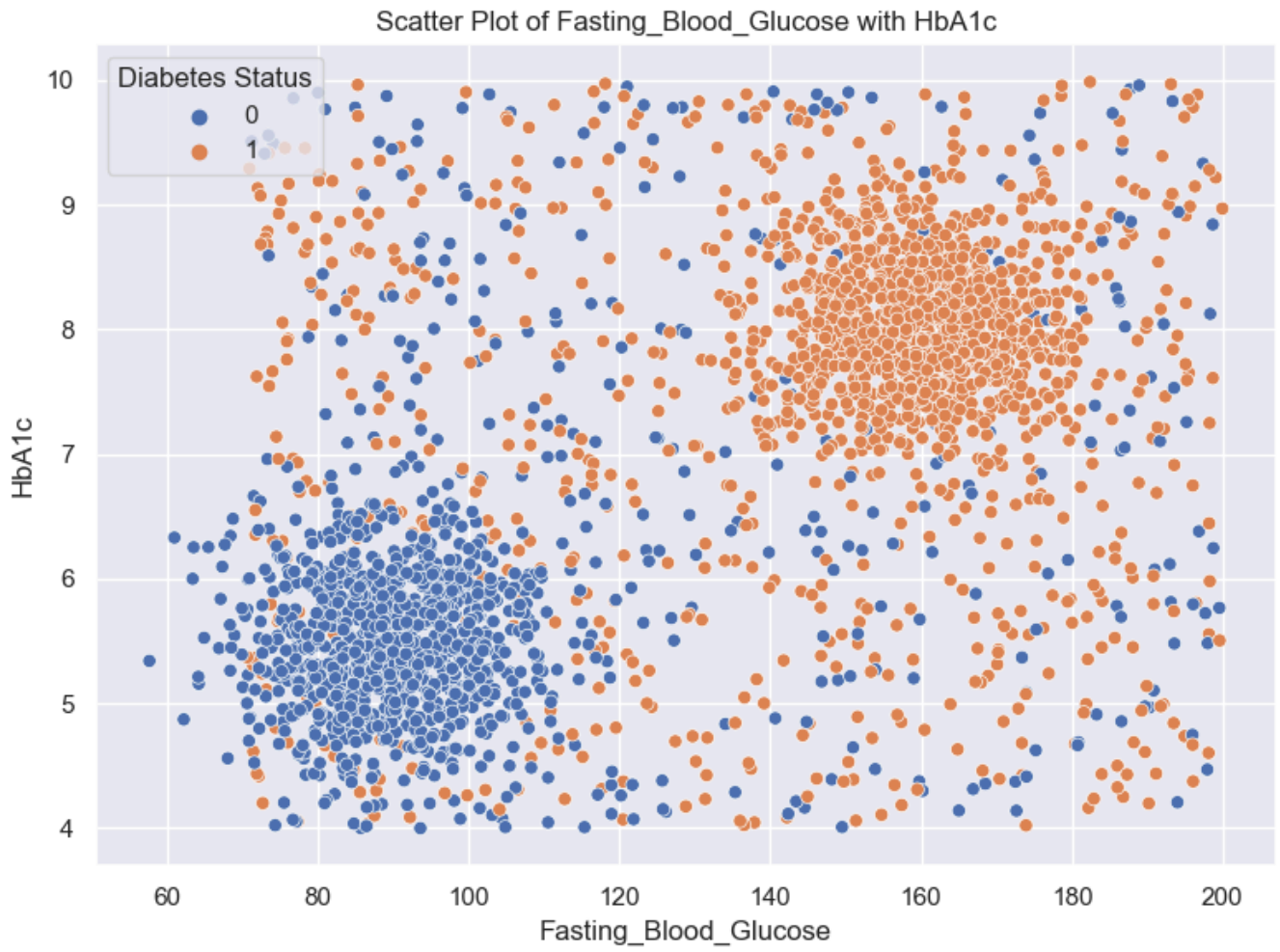
```
['CRP_Levels', 'Insulin_Levels'],
['CRP_Levels', 'HOMA_IR'],
['CRP_Levels', 'OGTT'],
['CRP_Levels', 'Creatinine_Levels'],
['CRP_Levels', 'eGFR'],
['CRP_Levels', 'Microalbuminuria'],
['CRP_Levels', 'Uric_Acid_Levels'],
['CRP_Levels', 'Fructosamine_Levels'],
['ALT', 'CRP_Levels'],
['AST', 'CRP_Levels'],
['CRP_Levels', 'C_Peptide'],
['CRP_Levels', 'Proinsulin_Levels'],
['HOMA_IR', 'Insulin_Levels'],
['Insulin_Levels', 'OGTT'],
['Creatinine_Levels', 'Insulin_Levels'],
['Insulin_Levels', 'eGFR'],
['Insulin_Levels', 'Microalbuminuria'],
['Insulin_Levels', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'Insulin_Levels'],
['ALT', 'Insulin_Levels'],
['AST', 'Insulin_Levels'],
['C_Peptide', 'Insulin_Levels'],
['Insulin_Levels', 'Proinsulin_Levels'],
['HOMA_IR', 'OGTT'],
['Creatinine_Levels', 'HOMA_IR'],
['HOMA_IR', 'eGFR'],
['HOMA_IR', 'Microalbuminuria'],
['HOMA_IR', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'HOMA_IR'],
['ALT', 'HOMA_IR'],
['AST', 'HOMA_IR'],
['C_Peptide', 'HOMA_IR'],
['HOMA_IR', 'Proinsulin_Levels'],
['Creatinine_Levels', 'OGTT'],
['OGTT', 'eGFR'],
['Microalbuminuria', 'OGTT'],
['OGTT', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'OGTT'],
['ALT', 'OGTT'],
['AST', 'OGTT'],
['C_Peptide', 'OGTT'],
['OGTT', 'Proinsulin_Levels'],
['Creatinine_Levels', 'eGFR'],
['Creatinine_Levels', 'Microalbuminuria'],
['Creatinine_Levels', 'Uric_Acid_Levels'],
['Creatinine_Levels', 'Fructosamine_Levels'],
['ALT', 'Creatinine_Levels'],
['AST', 'Creatinine_Levels'],
['C_Peptide', 'Creatinine_Levels'],
['Creatinine_Levels', 'Proinsulin_Levels'],
['Microalbuminuria', 'eGFR'],
['Uric_Acid_Levels', 'eGFR'],
['Fructosamine_Levels', 'eGFR'],
['ALT', 'eGFR'],
['AST', 'eGFR'],
['C_Peptide', 'eGFR'],
['Proinsulin_Levels', 'eGFR'],
['Microalbuminuria', 'Uric_Acid_Levels'],
['Fructosamine_Levels', 'Microalbuminuria'],
['ALT', 'Microalbuminuria'],
['AST', 'Microalbuminuria'],
['C_Peptide', 'Microalbuminuria'],
['Microalbuminuria', 'Proinsulin_Levels'],
['Fructosamine_Levels', 'Uric_Acid_Levels'],
['ALT', 'Uric_Acid_Levels'],
['AST', 'Uric_Acid_Levels'],
```

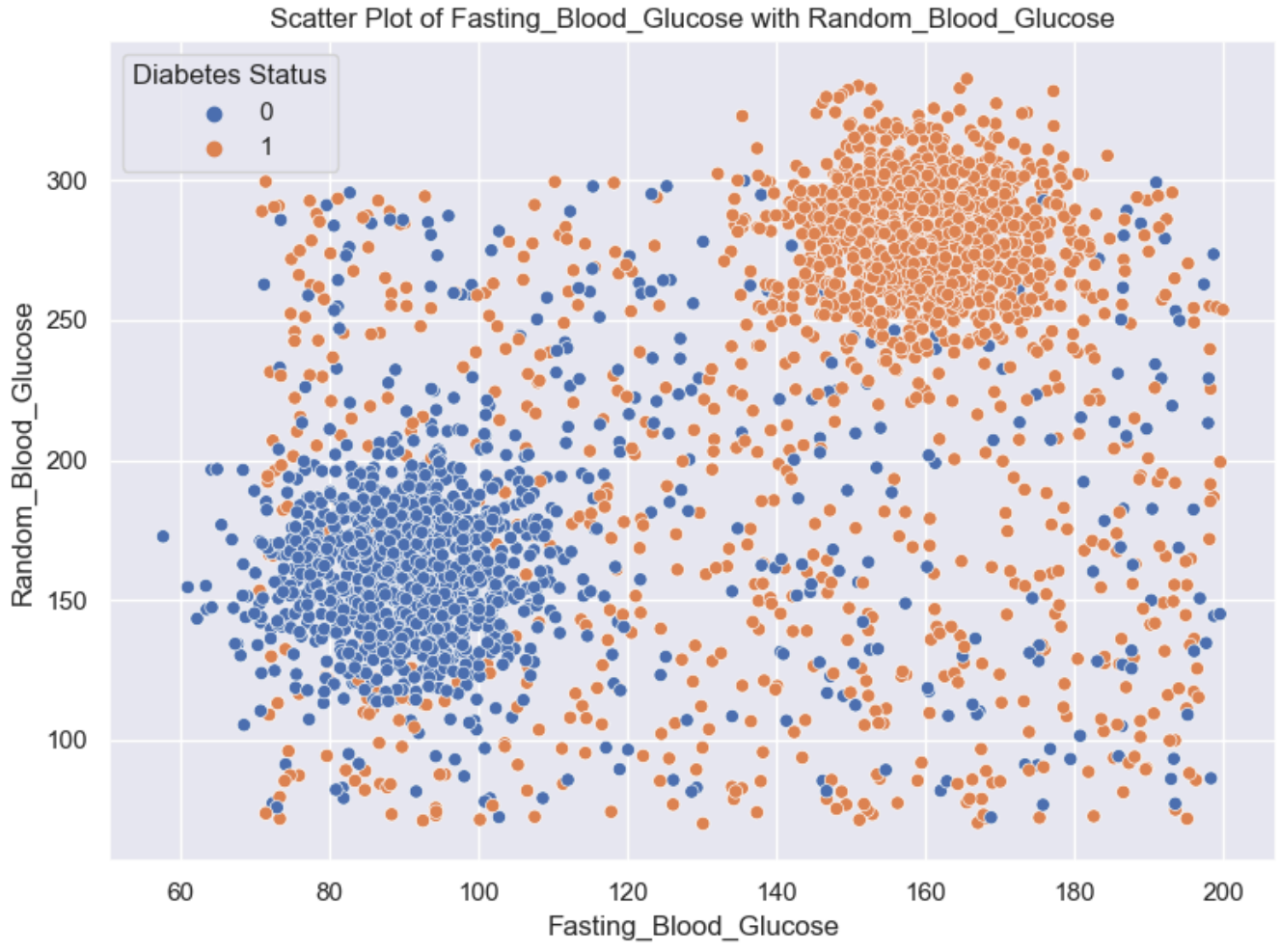
```
['C_Peptide', 'Uric_Acid_Levels'],
['Proinsulin_Levels', 'Uric_Acid_Levels'],
['ALT', 'Fructosamine_Levels'],
['AST', 'Fructosamine_Levels'],
['C_Peptide', 'Fructosamine_Levels'],
['Fructosamine_Levels', 'Proinsulin_Levels'],
['ALT', 'AST'],
['ALT', 'C_Peptide'],
['ALT', 'Proinsulin_Levels'],
['AST', 'C_Peptide'],
['AST', 'Proinsulin_Levels'],
['C_Peptide', 'Proinsulin_Levels']]
```

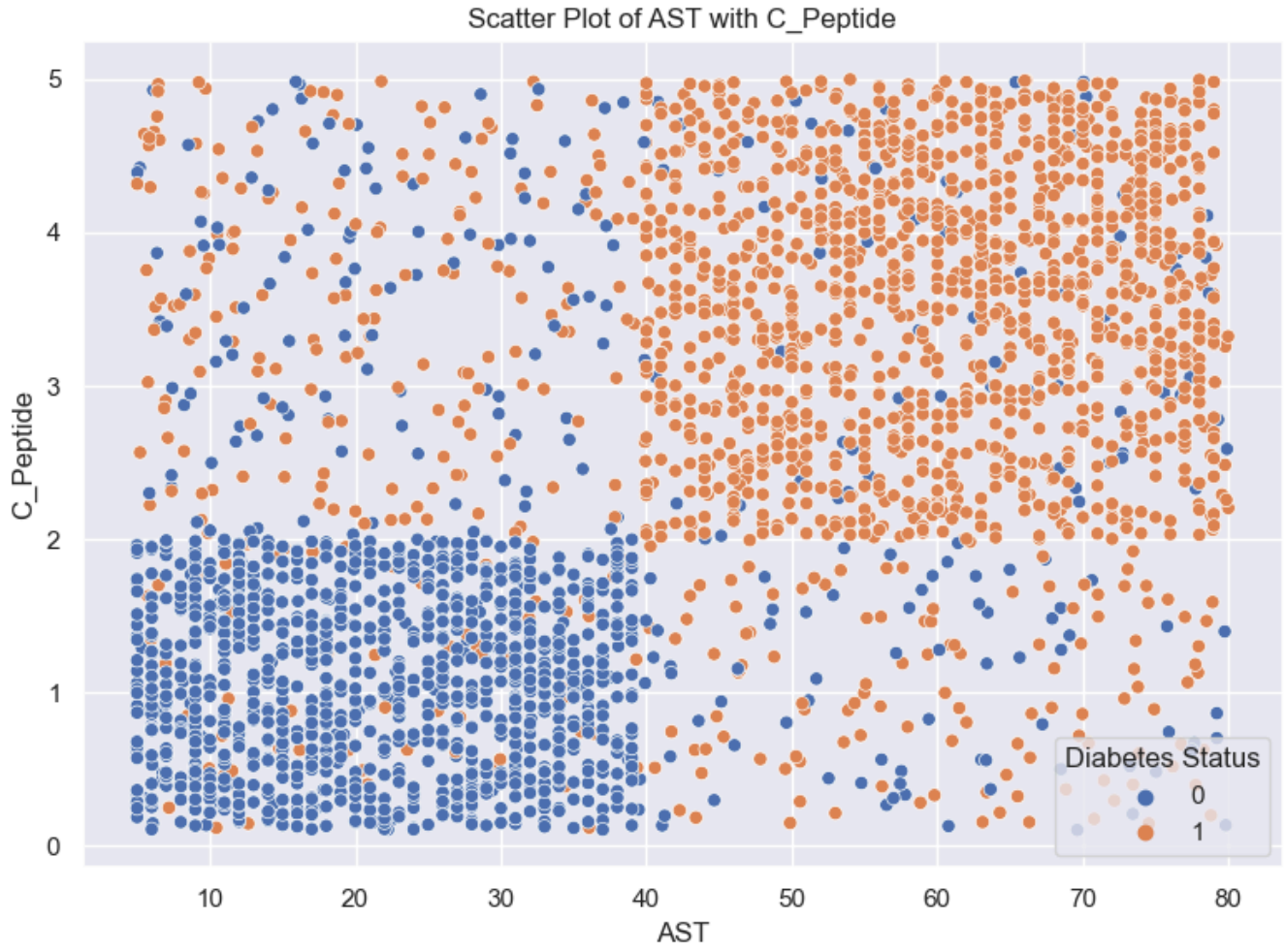
In [16]:

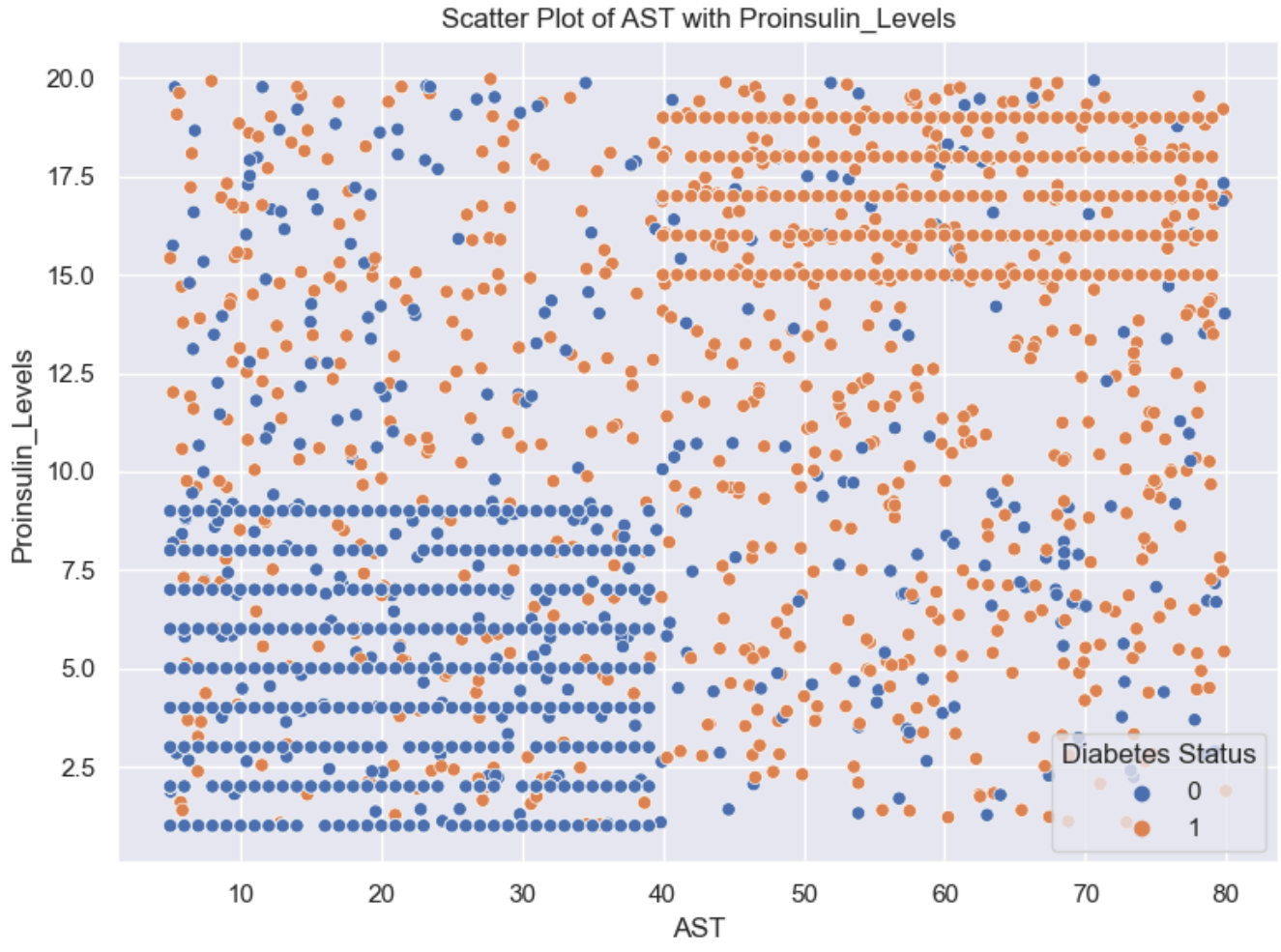
```python
for col1, col2 in lst:
    plt.figure(figsize=(8,6))
    sns.scatterplot(data= df, x=col1, y=col2, hue='Diabetes_Status')
    plt.title(f'Scatter Plot of {col1} with {col2}')
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.legend(title='Diabetes Status')
    plt.tight_layout()
    plt.show()
```

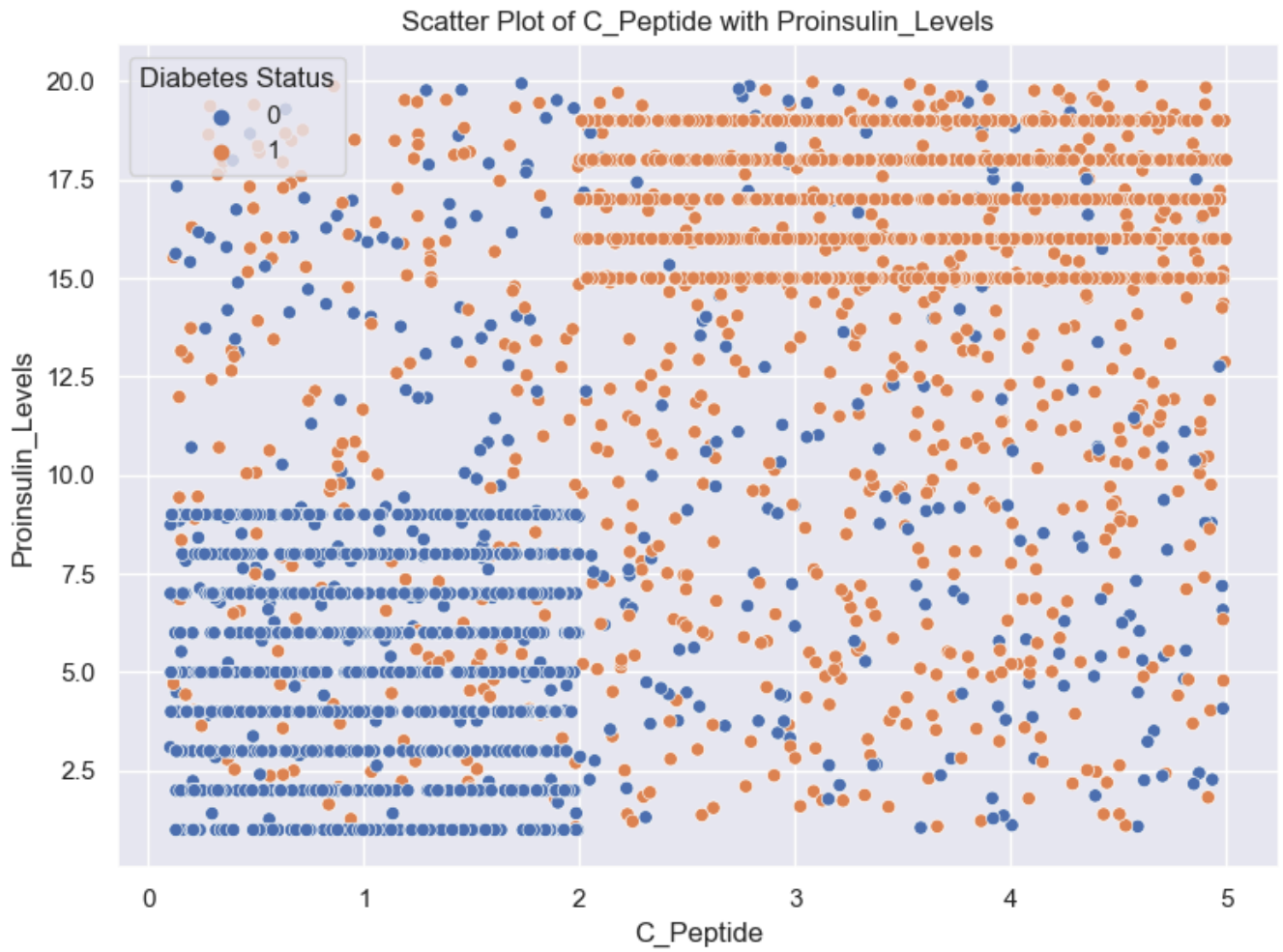



Scatter Plot of Fasting_Blood_Glucose with HbA1c
Diabetes Status
0
1
HbA1c
10
9
8
7
6
5
4
60
80
100
120
140
160
180
200
Fasting_Blood_Glucose


Scatter Plot of Fasting_Blood_Glucose with Random_Blood_Glucose
Diabetes Status
0
1
Random_Blood_Glucose
300
250
200
150
100
60
80
100
120
140
160
180
200
Fasting_Blood_Glucose

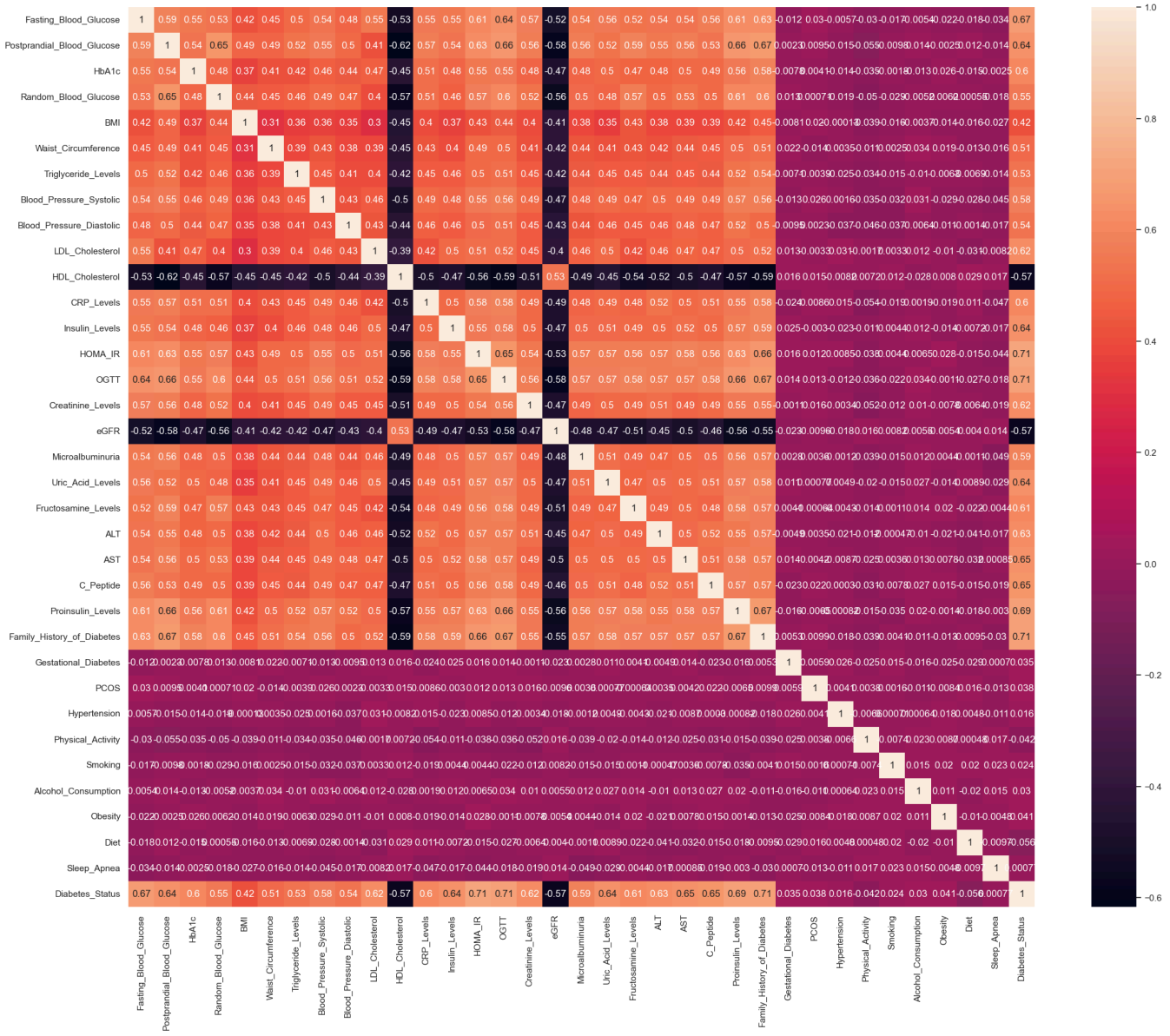

Scatter Plot of AST with C_Peptide
C_Peptide
AST
Diabetes Status
0
1


Scatter Plot of AST with Proinsulin_Levels
Proinsulin_Levels
AST
Diabetes Status
0
1



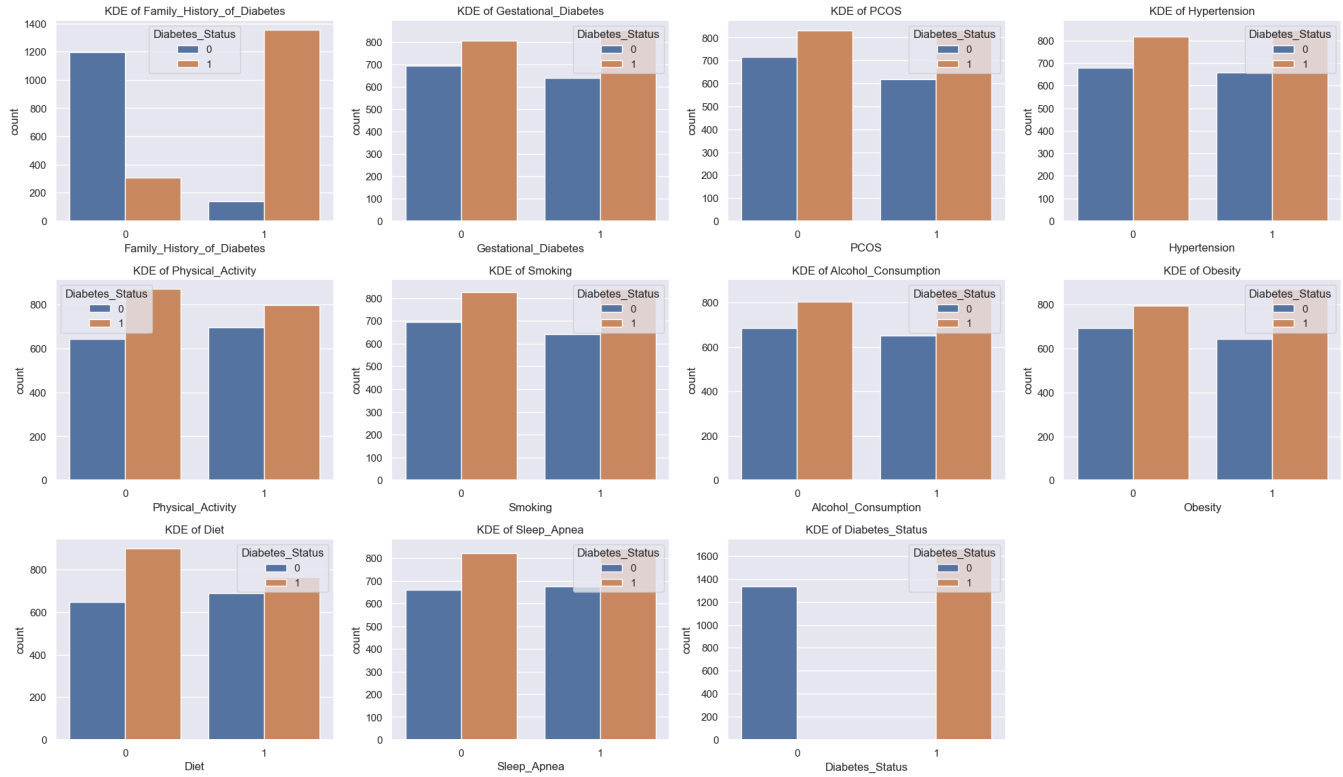
Correlation between all the features

In [17]:
```
plt.figure(figsize=(25, 20))
# Using seaborn to create a heatmap for the correlation matrix
p = sns.heatmap(df.corr(), annot=True)
```



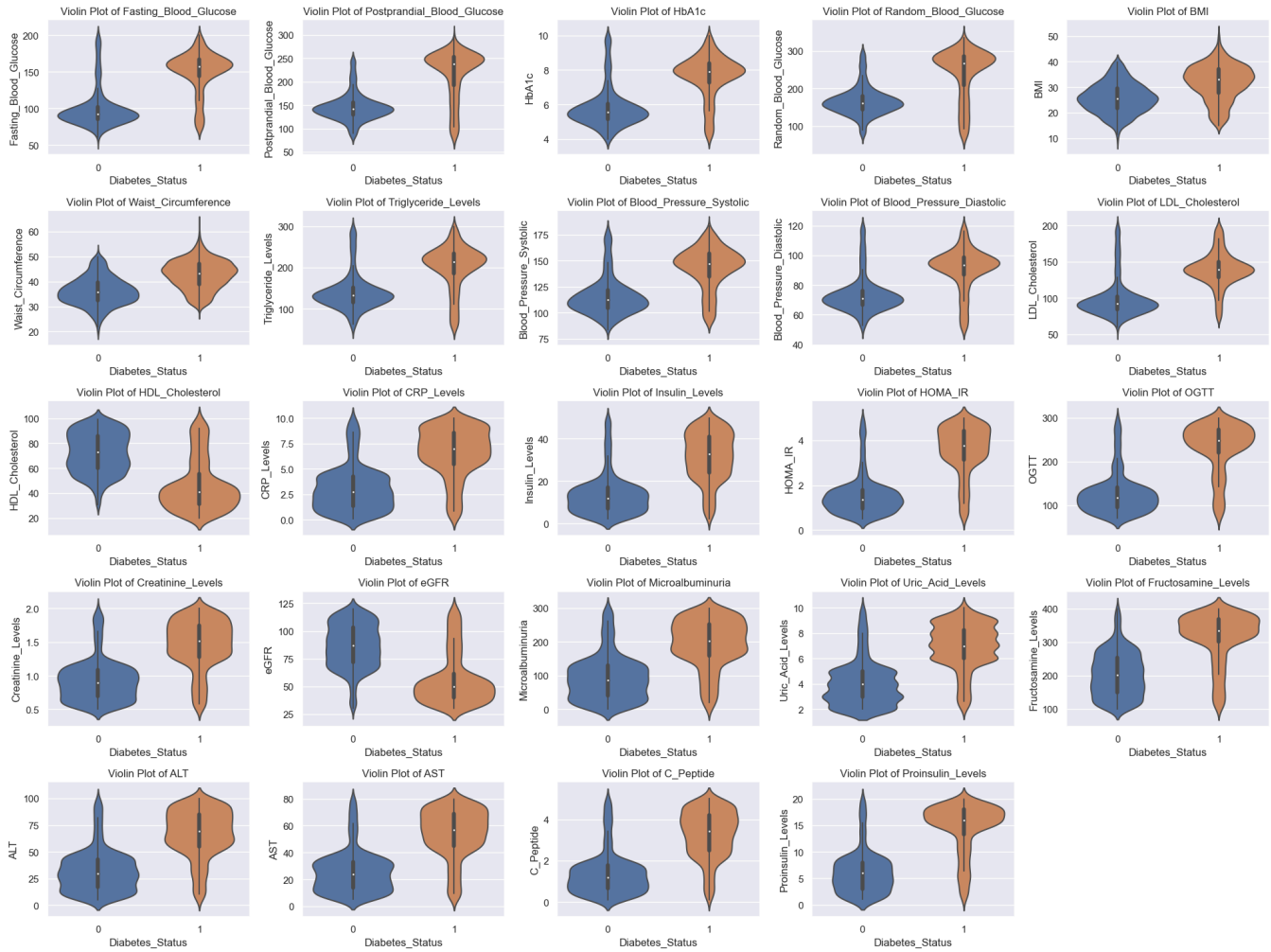
In [18]:

```
plt.figure(figsize=(20, 15))

cat_columns = ["Family_History_of_Diabetes","Gestational_Diabetes","PCOS","Hypertension","Phy

for i, col in enumerate(cat_columns):
    plt.subplot(4, 4, i+1)
    sns.countplot(data=df, x=col, hue='Diabetes_Status', fill=True)
    plt.title(f'KDE of {col}')
plt.tight_layout()
plt.show()
```



In [19]:

```
plt.figure(figsize=(20, 15))
for i, col in enumerate(numeric_columns):
    plt.subplot(5, 5, i+1)
    sns.violinplot(x='Diabetes_Status', y=col, data=df)
    plt.title(f'Violin Plot of {col}')
plt.tight_layout()
plt.show()
```



Model Building And Selection

In [20]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curv

def model_score(model, y_test, y_pred, y_prob=None, plot_roc=True):
    print('Classification Algorithm:', model)
    print('\nClassification Report:\n', classification_report(y_test, y_pred))
    print('\nConfusion Matrix:\n', confusion_matrix(y_test, y_pred))
    print(f'\nAccuracy Score: {accuracy_score(y_test, y_pred):.4f}')
#     auc_value = roc_auc_score(y_test, y_prob)
#     print(f'AUC: {auc_value:.4f}')

    if plot_roc and y_prob is not None:
        fpr, tpr, thresholds = roc_curve(y_test, y_prob)
        auc_value = roc_auc_score(y_test, y_prob)

        plt.figure(figsize=(8, 6))
        plt.plot(fpr, tpr, color='blue', label=f'AUC = {auc_value:.2f}')
        plt.plot([0, 1], [0, 1], color='red', linestyle='--')
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver Operating Characteristic (ROC) Curve')
        plt.legend(loc='lower right')
        plt.show()
        print(f'AUC: {auc_value:.4f}')
```

Logistic Regression

In [21]:
```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
lr_model.fit(X_train_scaled,y_train)
lr_pred = lr_model.predict(X_test_scaled)
lr_prob = lr_model.predict_proba(X_test)[:, 1]


model_score(lr_model,y_test,lr_pred)
```

```
Classification Algorithm: LogisticRegression()

Classification Report:
               precision    recall  f1-score   support

           0       0.93      0.87      0.90       271
           1       0.90      0.95      0.92       329

    accuracy                           0.91       600
   macro avg       0.91      0.91      0.91       600
weighted avg       0.91      0.91      0.91       600


Confusion Matrix:
 [[236  35]
 [ 18 311]]

Accuracy Score: 0.9117
```

In [22]:
```
from sklearn.model_selection import cross_val_score, train_test_split
lr_model = LogisticRegression(max_iter=1000)

scores = cross_val_score(lr_model, X_train_scaled, y_train, cv=5, scoring='accuracy')  # Exam

# Print cross-validation scores
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {scores.mean():.4f}')
```

```
Cross-Validation Scores: [0.92708333 0.92708333 0.92916667 0.91875    0.9375    ]
Mean Accuracy: 0.9279
```

Support Vector Machine

In [23]:
```python
from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train_scaled,y_train)
sv_pred = svc_model.predict(X_test_scaled)

model_score(svc_model,y_test,sv_pred)
```

```
Classification Algorithm: SVC()

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.80      0.87       271
           1       0.86      0.97      0.91       329

    accuracy                           0.89       600
   macro avg       0.90      0.89      0.89       600
weighted avg       0.90      0.89      0.89       600


Confusion Matrix:
 [[218  53]
 [ 11 318]]

Accuracy Score: 0.8933
```

K Nearest Neighbors

In [24]:
```python
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,y_train)
knn_pred = knn_model.predict(X_test)

model_score(knn_model, y_test, knn_pred)
```

```
Classification Algorithm: KNeighborsClassifier()

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.79      0.82       271
           1       0.84      0.90      0.87       329

    accuracy                           0.85       600
   macro avg       0.85      0.84      0.85       600
weighted avg       0.85      0.85      0.85       600


Confusion Matrix:
 [[213  58]
 [ 33 296]]

Accuracy Score: 0.8483
```

Random Forest

In [25]:
```python
from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
rfc_pred = rfc.predict(X_test)
model_score(rfc, y_test, rfc_pred)
```

```
Classification Algorithm: RandomForestClassifier(n_estimators=200)

Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.76      0.85       271
           1       0.83      0.98      0.90       329

    accuracy                           0.88       600
   macro avg       0.90      0.87      0.88       600
weighted avg       0.90      0.88      0.88       600


Confusion Matrix:
 [[205  66]
 [  5 324]]

Accuracy Score: 0.8817
```

Decision Tree

In [26]:
```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_test)

model_score(dt_model, y_test, dt_pred)
```

```
Classification Algorithm: DecisionTreeClassifier()

Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.83      0.84       271
           1       0.86      0.89      0.88       329

    accuracy                           0.86       600
   macro avg       0.86      0.86      0.86       600
weighted avg       0.86      0.86      0.86       600


Confusion Matrix:
 [[224  47]
 [ 36 293]]

Accuracy Score: 0.8617
```

In [27]:
```
from xgboost import XGBClassifier

xgb_model = XGBClassifier()
xgb_model.fit(X_train,y_train)
xgb_pred = xgb_model.predict(X_test)

model_score(xgb_model, y_test, xgb_pred)
```

```
Classification Algorithm: XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)

Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.87      0.91       271
           1       0.90      0.96      0.93       329

    accuracy                           0.92       600
   macro avg       0.92      0.92      0.92       600
weighted avg       0.92      0.92      0.92       600


Confusion Matrix:
 [[235  36]
 [ 12 317]]

Accuracy Score: 0.9200
```

Neural Network

In [28]:
```
import tensorflow as tf
```

In [29]:
```
model_nn  = tf.keras.Sequential([
    tf.keras.layers.Dense(32, input_shape = (X_train.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1,  activation='sigmoid')
])

model_nn.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

history = model_nn.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_data=(X_
```

```
Epoch 1/50
75/75 [==============================] - 1s 6ms/step - loss: 0.2795 - accuracy: 0.8954 - val_
loss: 0.2019 - val_accuracy: 0.9017
Epoch 2/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1511 - accuracy: 0.9304 - val_
loss: 0.1874 - val_accuracy: 0.9083
Epoch 3/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1338 - accuracy: 0.9337 - val_
loss: 0.1803 - val_accuracy: 0.9133
Epoch 4/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1241 - accuracy: 0.9404 - val_
loss: 0.1783 - val_accuracy: 0.9167
Epoch 5/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1158 - accuracy: 0.9475 - val_
loss: 0.1840 - val_accuracy: 0.9100
Epoch 6/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1085 - accuracy: 0.9463 - val_
loss: 0.1830 - val_accuracy: 0.9150
Epoch 7/50
75/75 [==============================] - 0s 2ms/step - loss: 0.1031 - accuracy: 0.9550 - val_
loss: 0.1840 - val_accuracy: 0.9133
Epoch 8/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0966 - accuracy: 0.9617 - val_
loss: 0.1910 - val_accuracy: 0.9067
Epoch 9/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0893 - accuracy: 0.9646 - val_
loss: 0.1892 - val_accuracy: 0.9133
Epoch 10/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0839 - accuracy: 0.9683 - val_
loss: 0.1975 - val_accuracy: 0.9133
Epoch 11/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0766 - accuracy: 0.9700 - val_
loss: 0.2005 - val_accuracy: 0.9117
Epoch 12/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0704 - accuracy: 0.9762 - val_
loss: 0.1950 - val_accuracy: 0.9167
Epoch 13/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0646 - accuracy: 0.9754 - val_
loss: 0.1987 - val_accuracy: 0.9183
Epoch 14/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0580 - accuracy: 0.9800 - val_
loss: 0.2054 - val_accuracy: 0.9150
Epoch 15/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0521 - accuracy: 0.9842 - val_
loss: 0.2131 - val_accuracy: 0.9167
Epoch 16/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0476 - accuracy: 0.9842 - val_
loss: 0.2225 - val_accuracy: 0.9167
Epoch 17/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0431 - accuracy: 0.9879 - val_
loss: 0.2272 - val_accuracy: 0.9183
Epoch 18/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0382 - accuracy: 0.9900 - val_
loss: 0.2469 - val_accuracy: 0.9067
Epoch 19/50
75/75 [==============================] - 0s 3ms/step - loss: 0.0341 - accuracy: 0.9904 - val_
loss: 0.2447 - val_accuracy: 0.9183
Epoch 20/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0306 - accuracy: 0.9908 - val_
loss: 0.2611 - val_accuracy: 0.9083
Epoch 21/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0258 - accuracy: 0.9954 - val_
loss: 0.2759 - val_accuracy: 0.9167
Epoch 22/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0224 - accuracy: 0.9962 - val_
loss: 0.2847 - val_accuracy: 0.9183
```

```
Epoch 23/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0193 - accuracy: 0.9954 - val_
loss: 0.3052 - val_accuracy: 0.9117
Epoch 24/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0172 - accuracy: 0.9975 - val_
loss: 0.3099 - val_accuracy: 0.9217
Epoch 25/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0133 - accuracy: 0.9983 - val_
loss: 0.3231 - val_accuracy: 0.9167
Epoch 26/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0121 - accuracy: 0.9987 - val_
loss: 0.3340 - val_accuracy: 0.9167
Epoch 27/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0099 - accuracy: 1.0000 - val_
loss: 0.3736 - val_accuracy: 0.9133
Epoch 28/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0086 - accuracy: 1.0000 - val_
loss: 0.3712 - val_accuracy: 0.9183
Epoch 29/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0068 - accuracy: 1.0000 - val_
loss: 0.3846 - val_accuracy: 0.9183
Epoch 30/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0057 - accuracy: 1.0000 - val_
loss: 0.4038 - val_accuracy: 0.9183
Epoch 31/50
75/75 [==============================] - 0s 3ms/step - loss: 0.0050 - accuracy: 1.0000 - val_
loss: 0.4118 - val_accuracy: 0.9183
Epoch 32/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0043 - accuracy: 1.0000 - val_
loss: 0.4265 - val_accuracy: 0.9167
Epoch 33/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0039 - accuracy: 1.0000 - val_
loss: 0.4324 - val_accuracy: 0.9167
Epoch 34/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0033 - accuracy: 1.0000 - val_
loss: 0.4486 - val_accuracy: 0.9183
Epoch 35/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0028 - accuracy: 1.0000 - val_
loss: 0.4624 - val_accuracy: 0.9150
Epoch 36/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0026 - accuracy: 1.0000 - val_
loss: 0.4833 - val_accuracy: 0.9133
Epoch 37/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0024 - accuracy: 1.0000 - val_
loss: 0.4855 - val_accuracy: 0.9150
Epoch 38/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0020 - accuracy: 1.0000 - val_
loss: 0.4850 - val_accuracy: 0.9167
Epoch 39/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0018 - accuracy: 1.0000 - val_
loss: 0.5070 - val_accuracy: 0.9150
Epoch 40/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0016 - accuracy: 1.0000 - val_
loss: 0.5151 - val_accuracy: 0.9117
Epoch 41/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0015 - accuracy: 1.0000 - val_
loss: 0.5179 - val_accuracy: 0.9183
Epoch 42/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0013 - accuracy: 1.0000 - val_
loss: 0.5303 - val_accuracy: 0.9117
Epoch 43/50
75/75 [==============================] - 0s 2ms/step - loss: 0.0012 - accuracy: 1.0000 - val_
loss: 0.5382 - val_accuracy: 0.9150
Epoch 44/50
75/75 [==============================] - 0s 1ms/step - loss: 0.0011 - accuracy: 1.0000 - val_
loss: 0.5530 - val_accuracy: 0.9167
```

```
Epoch 45/50
75/75 [==============================] - 0s 2ms/step - loss: 9.5335e-04 - accuracy: 1.0000 -
val_loss: 0.5573 - val_accuracy: 0.9167
Epoch 46/50
75/75 [==============================] - 0s 2ms/step - loss: 8.5118e-04 - accuracy: 1.0000 -
val_loss: 0.5573 - val_accuracy: 0.9150
Epoch 47/50
75/75 [==============================] - 0s 2ms/step - loss: 7.9759e-04 - accuracy: 1.0000 -
val_loss: 0.5722 - val_accuracy: 0.9150
Epoch 48/50
75/75 [==============================] - 0s 2ms/step - loss: 7.2994e-04 - accuracy: 1.0000 -
val_loss: 0.5760 - val_accuracy: 0.9133
Epoch 49/50
75/75 [==============================] - 0s 2ms/step - loss: 6.7465e-04 - accuracy: 1.0000 -
val_loss: 0.5880 - val_accuracy: 0.9133
Epoch 50/50
75/75 [==============================] - 0s 2ms/step - loss: 6.3133e-04 - accuracy: 1.0000 -
val_loss: 0.5900 - val_accuracy: 0.9133
```

In [30]:
```
nn_pred = model_nn.predict(X_test_scaled)
```

```
19/19 [==============================] - 0s 645us/step
```

In [31]:
```
predictions = np.where(nn_pred > 0.5,1,0)
```

In [32]:
```
model_score(model_nn, y_test, predictions)
```

```
Classification Algorithm: <keras.engine.sequential.Sequential object at 0x000001856855EB50>

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.89      0.90       271
           1       0.91      0.94      0.92       329

    accuracy                           0.91       600
   macro avg       0.91      0.91      0.91       600
weighted avg       0.91      0.91      0.91       600


Confusion Matrix:
 [[240  31]
 [ 21 308]]

Accuracy Score: 0.9133
```

Cross Validation

In [33]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_
from sklearn.model_selection import cross_val_score, KFold


models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Support Vector Machine': SVC(probability=True),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'XGBoost': XGBClassifier(),

}

# Define metrics
metrics = {
    'Accuracy': 'accuracy',
    'Precision': 'precision_macro',
```

```
    'Recall': 'recall_macro',
    'F1 Score': 'f1_macro',
    'ROC AUC': 'roc_auc'
}

results = {}

# Perform cross-validation and compute metrics
for model_name, model in models.items():
    print(f'Calculating metrics for {model_name}...')
    cv = KFold(n_splits=5, shuffle=True, random_state=42)
    model_results = {}

    for metric_name, scoring in metrics.items():
        scores = cross_val_score(model, X_train_scaled, y_train, cv=cv, scoring=scoring)
        mean_score = np.mean(scores)
        model_results[metric_name] = mean_score

    results[model_name] = model_results

# Create a DataFrame to display results
results_df = pd.DataFrame(results).T
print("\nMean Scores Table:")
print(results_df)
```

```
Calculating metrics for Logistic Regression...
Calculating metrics for Support Vector Machine...
Calculating metrics for K-Nearest Neighbors...
Calculating metrics for Random Forest...
Calculating metrics for Decision Tree...
Calculating metrics for XGBoost...

Mean Scores Table:
                        Accuracy  Precision    Recall  F1 Score   ROC AUC
Logistic Regression     0.930000   0.930208  0.928062  0.928925  0.986238
Support Vector Machine  0.925000   0.928529  0.920620  0.923401  0.986520
K-Nearest Neighbors     0.918333   0.917960  0.916669  0.917168  0.969172
Random Forest           0.916667   0.927843  0.904495  0.911119  0.985220
Decision Tree           0.886667   0.878254  0.882027  0.884414  0.880906
XGBoost                 0.933750   0.935860  0.930333  0.932480  0.986647
```

In [34]:

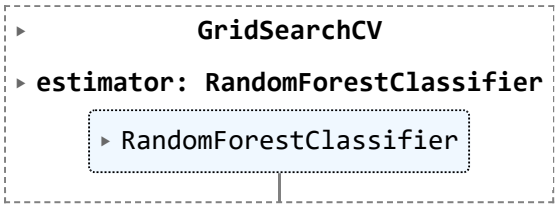
```
from sklearn.model_selection import GridSearchCV, train_test_split

rf = RandomForestClassifier()

rf_param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [50, 100],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

rf_grid_search = GridSearchCV(estimator=rf, param_grid=rf_param_grid, cv=3, scoring='accuracy

rf_grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.910
total time=   0.8s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.915
total time=   0.8s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.906
total time=   0.7s
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.916
total time=   1.6s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.910
total time=   1.6s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.907
total time=   1.5s
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.916
total time=   0.7s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.917
total time=   0.7s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.909
total time=   0.7s
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.910
total time=   1.5s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.915
total time=   1.5s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.919
total time=   1.5s
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.914
total time=   0.6s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.910
total time=   0.7s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.912
total time=   0.7s
[CV 1/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.909
total time=   1.5s
[CV 2/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.912
total time=   1.5s
[CV 3/3] END bootstrap=True, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.916
total time=   1.5s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.91
4 total time=   0.8s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.91
0 total time=   0.8s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.90
9 total time=   0.8s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.91
5 total time=   1.7s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.91
5 total time=   1.7s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.91
4 total time=   1.6s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.91
2 total time=   0.8s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.91
5 total time=   0.8s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.91
7 total time=   0.8s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.91
2 total time=   1.6s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.91
1 total time=   1.6s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.91
6 total time=   1.8s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.91
2 total time=   0.8s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.91
4 total time=   0.7s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.91
```

```
4 total time=   0.7s
[CV 1/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.91
7 total time=   1.8s
[CV 2/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.91
5 total time=   1.6s
[CV 3/3] END bootstrap=True, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.91
2 total time=   1.6s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.91
1 total time=   0.9s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.91
6 total time=   1.0s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=100;, score=0.92
4 total time=   0.9s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.92
0 total time=   2.2s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.91
9 total time=   2.0s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=1, n_estimators=200;, score=0.91
5 total time=   2.0s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.91
6 total time=   0.9s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.91
9 total time=   0.9s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=100;, score=0.92
1 total time=   0.9s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.91
5 total time=   1.8s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.92
4 total time=   1.9s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=2, n_estimators=200;, score=0.91
9 total time=   1.8s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.91
5 total time=   0.8s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.91
6 total time=   0.9s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=100;, score=0.91
7 total time=   0.7s
[CV 1/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.91
5 total time=   1.8s
[CV 2/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.91
6 total time=   1.8s
[CV 3/3] END bootstrap=False, max_depth=50, min_samples_leaf=4, n_estimators=200;, score=0.91
5 total time=   2.2s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.9
25 total time=   0.9s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.9
25 total time=   0.9s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=100;, score=0.9
15 total time=   1.0s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.9
21 total time=   1.9s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.9
24 total time=   2.1s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=1, n_estimators=200;, score=0.9
12 total time=   1.9s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.9
19 total time=   0.9s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.9
17 total time=   0.9s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=100;, score=0.9
17 total time=   0.8s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.9
21 total time=   1.8s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.9
20 total time=   2.1s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=2, n_estimators=200;, score=0.9
```

```
20 total time=   2.0s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.9
17 total time=   0.9s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.9
22 total time=   0.9s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=100;, score=0.9
17 total time=   0.9s
[CV 1/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.9
10 total time=   1.8s
[CV 2/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.9
14 total time=   1.9s
[CV 3/3] END bootstrap=False, max_depth=100, min_samples_leaf=4, n_estimators=200;, score=0.9
20 total time=   1.8s
```

Out[34]:



In [35]:

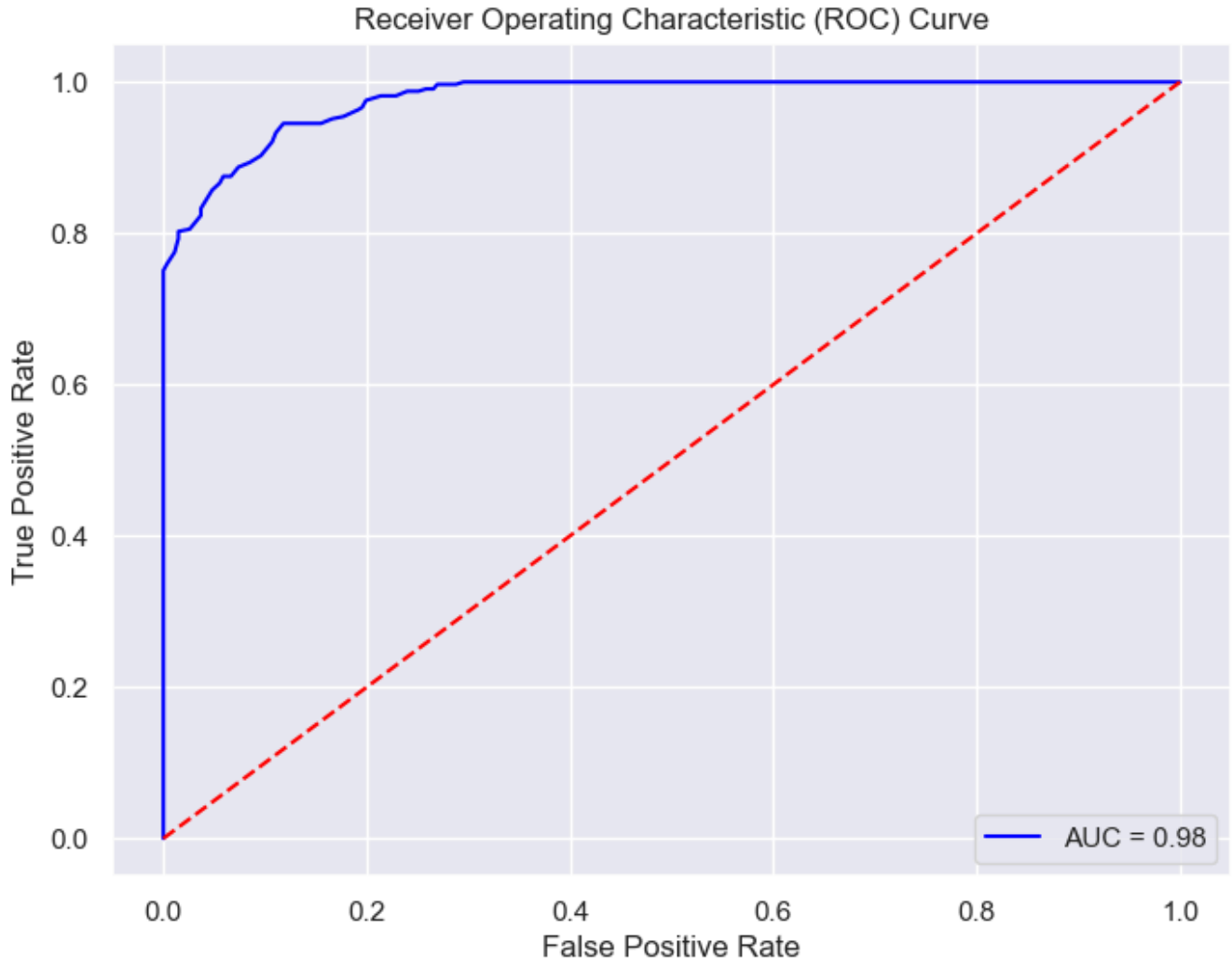
```
rf_best_params = rf_grid_search.best_params_
rf_best_score = rf_grid_search.best_score_

print("Random Forest Best Parameters:", rf_best_params)
print("Random Forest Best Cross-Validation Accuracy:", rf_best_score)

rf_best_model = rf_grid_search.best_estimator_
rf_prob = rf_best_model.predict_proba(X_test)[:,1]
y_pred_rf = rf_best_model.predict(X_test)
print("\nClassification Report on Test Data:")
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Best Parameters: {'bootstrap': False, 'max_depth': 100, 'min_samples_leaf': 1,
'n_estimators': 100}
Random Forest Best Cross-Validation Accuracy: 0.9216666666666667

Classification Report on Test Data:
              precision    recall  f1-score   support

           0       0.97      0.79      0.87       271
           1       0.85      0.98      0.91       329

    accuracy                           0.89       600
   macro avg       0.91      0.88      0.89       600
weighted avg       0.90      0.89      0.89       600
```

In [36]:

```
model_score(rf_best_model,y_test,y_pred_rf,rf_prob)
```

```
Classification Algorithm: RandomForestClassifier(bootstrap=False, max_depth=100)

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.79      0.87       271
           1       0.85      0.98      0.91       329

    accuracy                           0.89       600
   macro avg       0.91      0.88      0.89       600
weighted avg       0.90      0.89      0.89       600


Confusion Matrix:
 [[213  58]
 [  6 323]]

Accuracy Score: 0.8933
```




```
AUC: 0.9782
```

In [37]:

```python
xgb_clf = XGBClassifier()


xgb_param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'gamma': [0, 0.1, 0.2]
}

# Create GridSearchCV object
xgb_grid_search = GridSearchCV(estimator=xgb_clf, param_grid=xgb_param_grid, cv=3, scoring='a
```

```
# Fit GridSearchCV
xgb_grid_search.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 54 candidates, totalling 162 fits
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.911 total t
ime=   0.1s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.912 total t
ime=   0.1s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.899 total t
ime=   0.1s
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.912 total t
ime=   0.4s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.916 total t
ime=   0.5s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.910 total t
ime=   0.4s
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.900 total t
ime=   0.5s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.912 total t
ime=   0.5s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.910 total t
ime=   0.5s
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.906 total t
ime=   1.0s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.912 total t
ime=   1.1s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.914 total t
ime=   1.1s
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.901 total t
ime=   0.6s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.906 total t
ime=   0.8s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.907 total t
ime=   0.7s
[CV 1/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.912 total t
ime=   1.2s
[CV 2/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.912 total t
ime=   1.2s
[CV 3/3] END gamma=0, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.914 total t
ime=   1.0s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.929 total ti
me=   0.1s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.927 total ti
me=   0.1s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.927 total ti
me=   0.1s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.935 total ti
me=   0.3s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.943 total ti
me=   0.4s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.927 total ti
me=   0.3s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.917 total ti
me=   0.3s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.921 total ti
me=   0.4s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.920 total ti
me=   1.0s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.925 total ti
me=   0.9s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.930 total ti
me=   0.9s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.921 total ti
me=   0.8s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.917 total ti
me=   0.4s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.924 total ti
me=   0.4s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.922 total ti
```

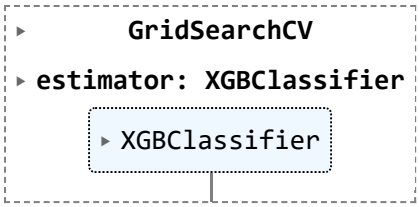
```
me=   0.5s
[CV 1/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.922 total ti
me=   0.6s
[CV 2/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.929 total ti
me=   0.6s
[CV 3/3] END gamma=0, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.926 total ti
me=   0.6s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.934 total ti
me=   0.1s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.932 total ti
me=   0.1s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.930 total ti
me=   0.1s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.934 total ti
me=   0.3s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.936 total ti
me=   0.3s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.930 total ti
me=   0.4s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.921 total ti
me=   0.2s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.926 total ti
me=   0.2s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.919 total ti
me=   0.2s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.925 total ti
me=   0.5s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.925 total ti
me=   0.5s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.920 total ti
me=   0.4s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.925 total ti
me=   0.3s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.922 total ti
me=   0.3s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.925 total ti
me=   0.3s
[CV 1/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.932 total ti
me=   0.7s
[CV 2/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.930 total ti
me=   0.8s
[CV 3/3] END gamma=0, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.927 total ti
me=   0.8s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.911 total
time=   0.8s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.912 total
time=   0.4s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.899 total
time=   0.3s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.912 total
time=   0.7s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.916 total
time=   0.5s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.910 total
time=   0.7s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.904 total
time=   0.4s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.910 total
time=   0.4s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.911 total
time=   0.4s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.906 total
time=   1.0s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.912 total
time=   0.8s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.910 total
```

```
time=   0.9s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.899 total
time=   0.5s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.906 total
time=   0.6s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.910 total
time=   0.7s
[CV 1/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.910 total
time=   1.9s
[CV 2/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.912 total
time=   1.2s
[CV 3/3] END gamma=0.1, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.914 total
time=   1.2s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.929 total
time=   0.2s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.927 total
time=   0.2s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.927 total
time=   0.2s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.932 total
time=   0.5s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.940 total
time=   0.4s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.925 total
time=   0.4s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.915 total
time=   0.3s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.924 total
time=   0.3s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.917 total
time=   0.3s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.920 total
time=   0.6s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.930 total
time=   0.6s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.926 total
time=   0.7s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.912 total
time=   0.4s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.921 total
time=   0.4s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.924 total
time=   0.4s
[CV 1/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.920 total
time=   0.6s
[CV 2/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.926 total
time=   0.7s
[CV 3/3] END gamma=0.1, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.924 total
time=   0.7s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.929 total
time=   0.2s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.932 total
time=   0.3s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.930 total
time=   0.2s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.936 total
time=   0.5s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.935 total
time=   0.4s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.929 total
time=   0.6s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.926 total
time=   0.3s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.925 total
time=   0.3s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.921 total
```

```
time=   0.4s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.926 total
time=   0.7s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.925 total
time=   0.7s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.921 total
time=   0.6s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.920 total
time=   0.4s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.926 total
time=   0.3s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.922 total
time=   0.4s
[CV 1/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.920 total
time=   0.5s
[CV 2/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.926 total
time=   0.5s
[CV 3/3] END gamma=0.1, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.922 total
time=   0.5s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.911 total
time=   0.2s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.912 total
time=   0.2s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=100;, score=0.899 total
time=   0.2s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.912 total
time=   0.5s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.916 total
time=   0.6s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=3, n_estimators=200;, score=0.910 total
time=   0.5s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.904 total
time=   0.8s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.906 total
time=   0.4s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=100;, score=0.907 total
time=   0.4s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.910 total
time=   0.9s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.911 total
time=   1.0s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=6, n_estimators=200;, score=0.914 total
time=   0.9s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.900 total
time=   0.5s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.905 total
time=   0.6s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=100;, score=0.912 total
time=   0.5s
[CV 1/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.907 total
time=   1.4s
[CV 2/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.912 total
time=   1.3s
[CV 3/3] END gamma=0.2, learning_rate=0.01, max_depth=9, n_estimators=200;, score=0.912 total
time=   1.2s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.930 total
time=   0.2s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.927 total
time=   0.2s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=100;, score=0.930 total
time=   0.2s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.932 total
time=   0.4s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.941 total
time=   0.6s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=3, n_estimators=200;, score=0.927 total
```

```
time=   0.6s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.921 total
time=   0.4s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.925 total
time=   0.4s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=100;, score=0.925 total
time=   0.4s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.921 total
time=   0.6s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.925 total
time=   0.7s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=6, n_estimators=200;, score=0.926 total
time=   0.7s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.921 total
time=   0.4s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.916 total
time=   0.4s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=100;, score=0.924 total
time=   0.4s
[CV 1/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.921 total
time=   0.6s
[CV 2/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.919 total
time=   0.7s
[CV 3/3] END gamma=0.2, learning_rate=0.1, max_depth=9, n_estimators=200;, score=0.922 total
time=   0.7s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.927 total
time=   0.2s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.930 total
time=   0.2s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=100;, score=0.925 total
time=   0.3s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.932 total
time=   0.5s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.931 total
time=   0.5s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=3, n_estimators=200;, score=0.924 total
time=   0.5s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.926 total
time=   0.3s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.926 total
time=   0.3s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=100;, score=0.920 total
time=   0.3s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.926 total
time=   0.5s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.926 total
time=   0.8s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=6, n_estimators=200;, score=0.920 total
time=   0.8s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.922 total
time=   0.5s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.922 total
time=   0.3s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=100;, score=0.917 total
time=   0.3s
[CV 1/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.922 total
time=   0.6s
[CV 2/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.922 total
time=   0.8s
[CV 3/3] END gamma=0.2, learning_rate=0.2, max_depth=9, n_estimators=200;, score=0.917 total
time=   0.6s
```

Out[37]:



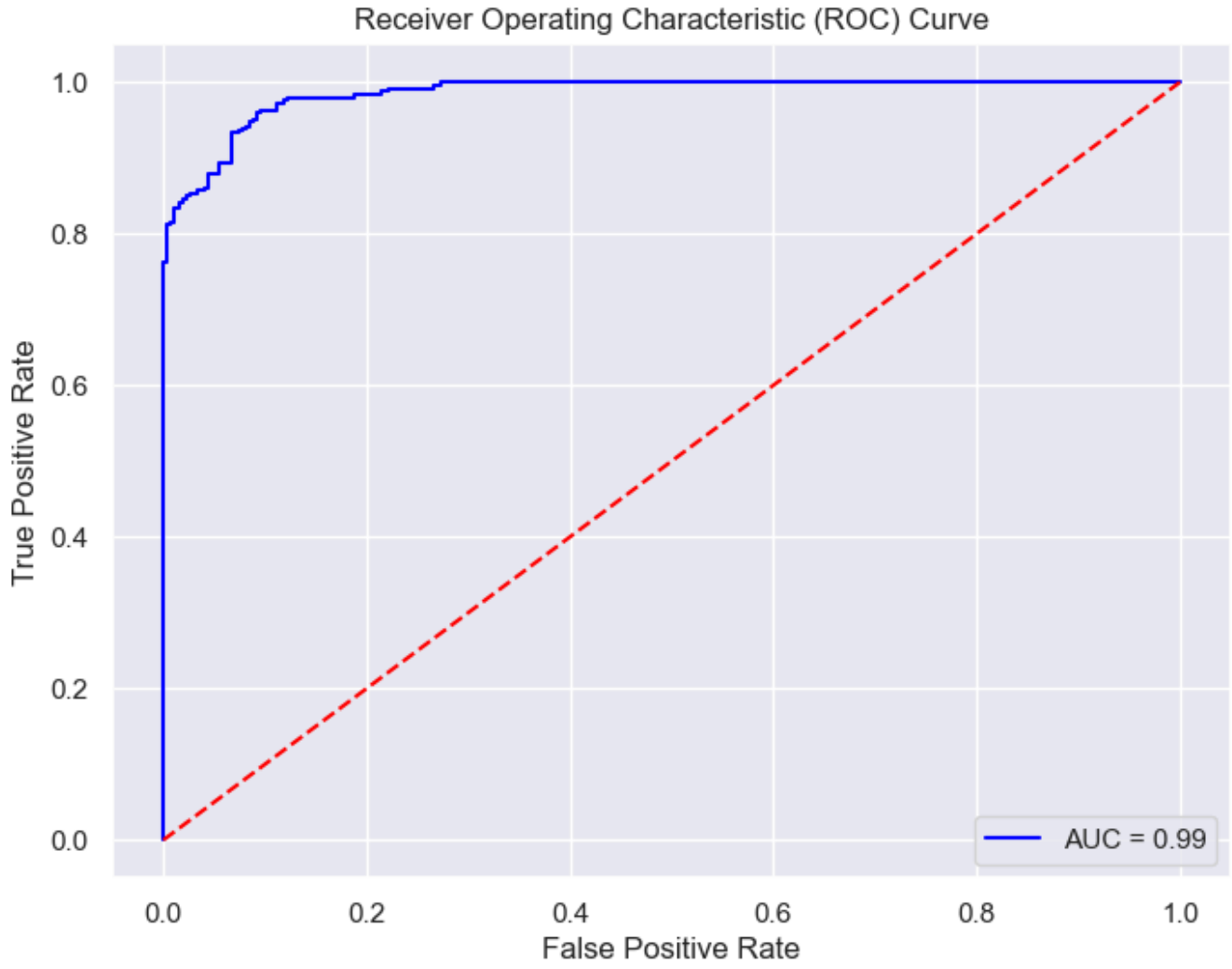
In [38]:
```
# Get the best parameters and score
xgb_best_params = xgb_grid_search.best_params_
xgb_best_score = xgb_grid_search.best_score_

# Print the results
print("XGBoost Best Parameters:", xgb_best_params)
print("XGBoost Best Cross-Validation Accuracy:", xgb_best_score)

# Evaluate on the test set
xgb_best_model = xgb_grid_search.best_estimator_
y_pred_xgb = xgb_best_model.predict(X_test)
print("\nClassification Report on Test Data:")
print(classification_report(y_test, y_pred_xgb))
```

```
XGBoost Best Parameters: {'gamma': 0, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 2
00}
XGBoost Best Cross-Validation Accuracy: 0.9349999999999999

Classification Report on Test Data:
              precision    recall  f1-score   support

           0       0.95      0.89      0.92       271
           1       0.91      0.96      0.94       329

    accuracy                           0.93       600
   macro avg       0.93      0.93      0.93       600
weighted avg       0.93      0.93      0.93       600
```

In [39]:
```
xgb_prob = xgb_best_model.predict_proba(X_test)[:,1]
```

In [40]:
```
model_score(xgb_best_model,y_test,y_pred_xgb,y_prob=xgb_prob)
```

```
Classification Algorithm: XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=200, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)

Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.89      0.92       271
           1       0.91      0.96      0.94       329

    accuracy                           0.93       600
   macro avg       0.93      0.93      0.93       600
weighted avg       0.93      0.93      0.93       600


Confusion Matrix:
 [[241  30]
 [ 12 317]]

Accuracy Score: 0.9300
```



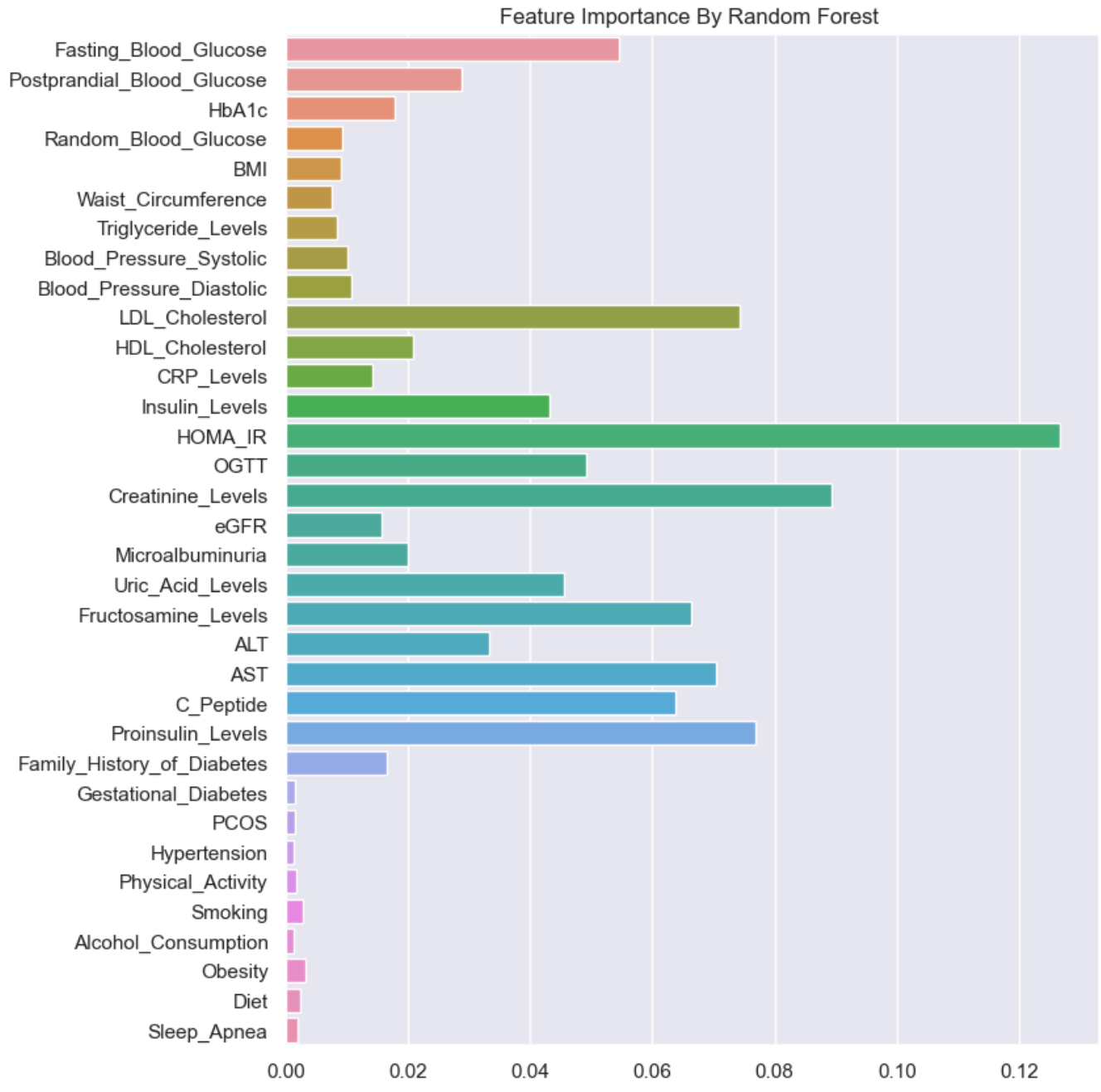

```
AUC: 0.9854
```

In [54]:
```
df.columns[:-1]
```

Out[54]:
```
Index(['Fasting_Blood_Glucose', 'Postprandial_Blood_Glucose', 'HbA1c',
       'Random_Blood_Glucose', 'BMI', 'Waist_Circumference',
       'Triglyceride_Levels', 'Blood_Pressure_Systolic',
       'Blood_Pressure_Diastolic', 'LDL_Cholesterol', 'HDL_Cholesterol',
       'CRP_Levels', 'Insulin_Levels', 'HOMA_IR', 'OGTT', 'Creatinine_Levels',
       'eGFR', 'Microalbuminuria', 'Uric_Acid_Levels', 'Fructosamine_Levels',
       'ALT', 'AST', 'C_Peptide', 'Proinsulin_Levels',
       'Family_History_of_Diabetes', 'Gestational_Diabetes', 'PCOS',
       'Hypertension', 'Physical_Activity', 'Smoking', 'Alcohol_Consumption',
       'Obesity', 'Diet', 'Sleep_Apnea'],
      dtype='object')
```

In [42]:
```
feature_values = rf_best_model.feature_importances_
```

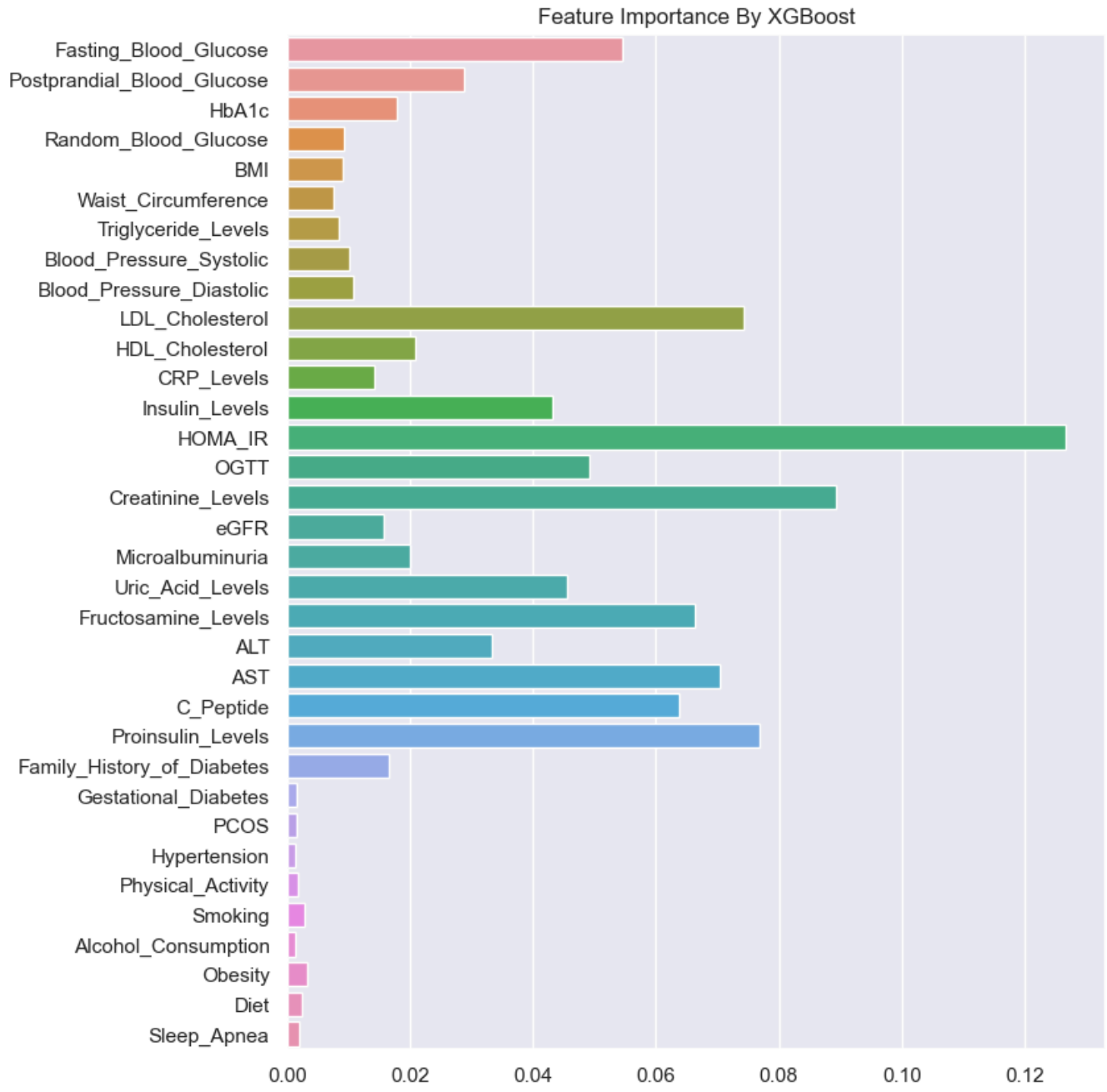
In [60]:
```
plt.figure(figsize=(8,10))
sns.barplot( x = feature_values,  y=df.columns[:-1] )
plt.title('Feature Importance By Random Forest ')
plt.show()
```



In [61]:
```
xgb_feature_values = xgb_best_model.feature_importances_
```

In [62]:
```
plt.figure(figsize=(8,10))
sns.barplot( x = feature_values,  y=df.columns[:-1] )
```

```
plt.title('Feature Importance By XGBoost')
plt.show()
```



Model Deployment

In [49]:
```
import pickle as pkl
```

In [52]:
```
pkl.dump(xgb_best_model,open('model_diabetes.pkl','wb'))
```

In [55]:
```
best_model = pkl.load(open('model_diabetes.pkl','rb'))
```

In [ ]:
```

```