In [1]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve
```

In [2]:
```
pip install mlxtend
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: mlxtend in c:\users\belea\appdata\roaming\pyth
on\python311\site-packages (0.23.1)
Requirement already satisfied: scipy>=1.2.1 in c:\programdata\anaconda3\lib\s
ite-packages (from mlxtend) (1.10.1)
Requirement already satisfied: numpy>=1.16.2 in c:\programdata\anaconda3\lib
\site-packages (from mlxtend) (1.24.3)
Requirement already satisfied: pandas>=0.24.2 in c:\programdata\anaconda3\lib
\site-packages (from mlxtend) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda
3\lib\site-packages (from mlxtend) (1.3.0)
Requirement already satisfied: matplotlib>=3.0.0 in c:\programdata\anaconda3
\lib\site-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in c:\programdata\anaconda3\lib
\site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib>=3.0.0->mlxtend) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anacond
a3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\s
ite-packages (from pandas>=0.24.2->mlxtend) (2022.7)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anacond
a3\lib\site-packages (from scikit-learn>=1.0.2->mlxtend) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install missingno
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: missingno in c:\users\belea\appdata\roaming\py
thon\python311\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (from missingno) (1.24.3)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\sit
e-packages (from missingno) (3.7.1)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-pac
kages (from missingno) (1.10.1)
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-p
ackages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib->missingno) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib->missingno) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib->missingno) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anacond
a3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pandas>=0.25 in c:\programdata\anaconda3\lib\s
ite-packages (from seaborn->missingno) (1.5.3)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\s
ite-packages (from pandas>=0.25->seaborn->missingno) (2022.7)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:\users\belea\appdata\roaming\pyth
on\python311\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.10.1)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
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
import missingno as msno
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
import sklearn.model_selection as mod
import sklearn.neighbors as nei
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import LeaveOneOut
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_au
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import RFECV
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve
import operator
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [6]:
```
df= pd.read_csv(r"C:\Users\belea\Downloads\diabetes.csv");df
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	.
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns

In [7]:
```
df.head()
```

Out[7]:

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

In [8]:
```
df.tail()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

In [9]:
```
# Exploratory data asnalysis
```

In [10]:
```
#number of rowsand coloumns in this dataset
df.shape
```

Out[10]:
```
(768, 9)
```

In [11]:
```
df.dtypes
```

Out[11]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [12]:
```
#information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [13]:
```
#Statistical measures of the data
df.describe()
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesI
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [14]:
```
#Feature and Target
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [16]:
```
#Duplicate values
```

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
0
```

In [18]:
```
#Missing values
```

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [20]:
```
#Imputation
```

In [21]:
```
df_copy =df.copy(deep=True)
```

In [22]:
```
df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df_copy[
```

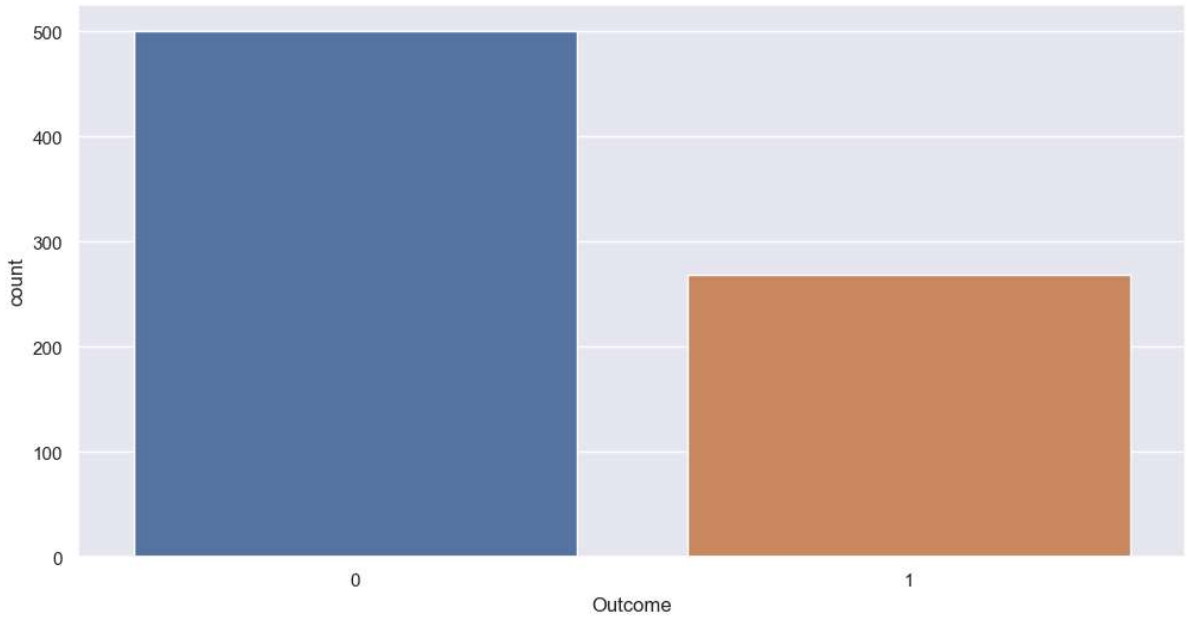
In [23]:
```
df_copy.isnull().sum()
```

Out[23]:
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [24]:
```
df['Outcome'].value_counts()
```

Out[24]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [25]:
```
plt.figure(figsize=(12,6))
sns.countplot(x='Outcome',data=df)
plt.show()
```



In [26]:
```
#0->Non-Diabetic
#1->Diabetic
```

In [27]:
```
#Imbalanced Dataset
```

In [28]:
```
df.groupby('Outcome').mean()
```

Out[28]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	

In [29]:
```
#Replace NaN Values
```

In [30]:
```
df1=df_copy
```

In [31]:
```
# imputing the mean value of the column to each missing value of that particula
```

In [32]:
```
df1['Glucose'].fillna(df1['Glucose'].mean(),inplace=True)
```

In [33]:
```
df1['BloodPressure'].fillna(df1['BloodPressure'].mean(),inplace=True)
```

In [34]:
```
df1['SkinThickness'].fillna(df1['SkinThickness'].median(),inplace=True)
```

In [35]:
```
df1['Insulin'].fillna(df1['Insulin'].median(),inplace=True)
```

In [36]:
```
df1['BMI'].fillna(df1['BMI'].median(),inplace=True)
```

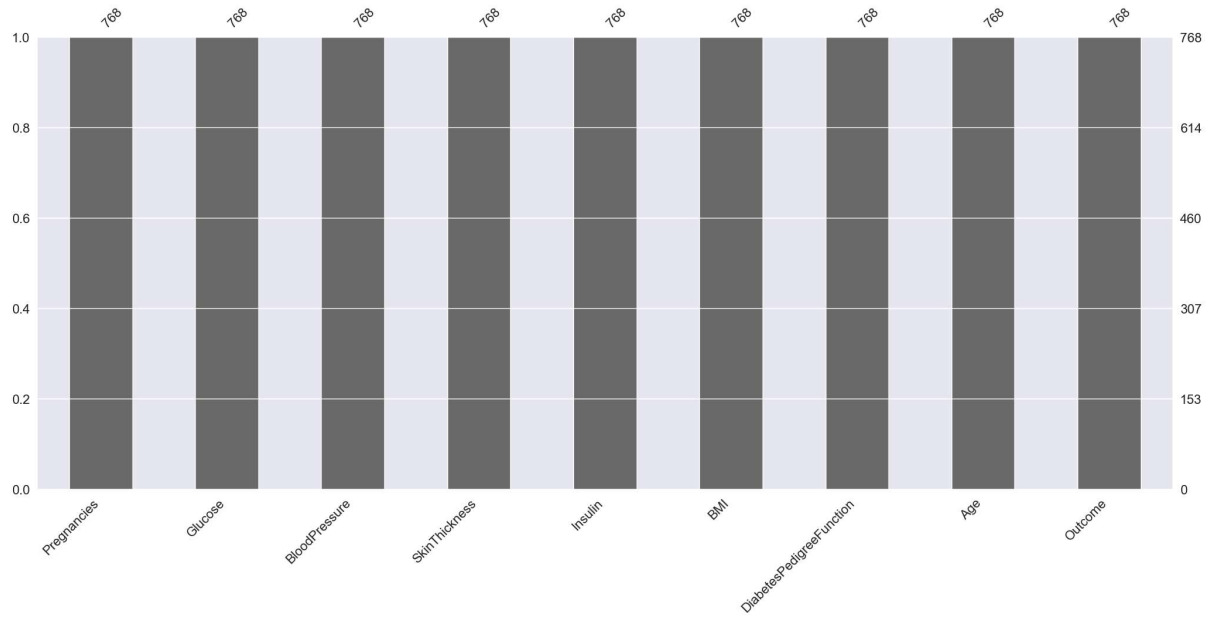
In [37]:
```
df1.isnull().sum()
```

Out[37]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [38]:
```
#Plotting Null Count Analysis Plot
```

In [39]:
```
msno.bar(df1)
```

Out[39]:
```
<Axes: >
```

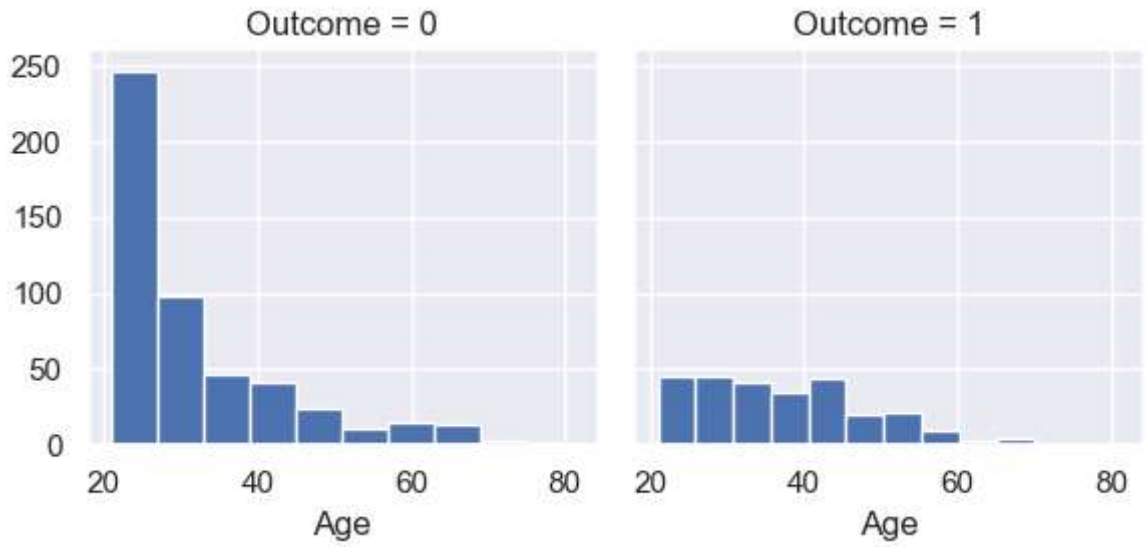


In [40]:
```
#Univariate EDA
```

In [41]:
```
#AGE
```

In [42]:
```
age=sns.FacetGrid(df1,col='Outcome')
age.map(plt.hist,'Age')
```

Out[42]:
```
<seaborn.axisgrid.FacetGrid at 0x16830a14610>
```



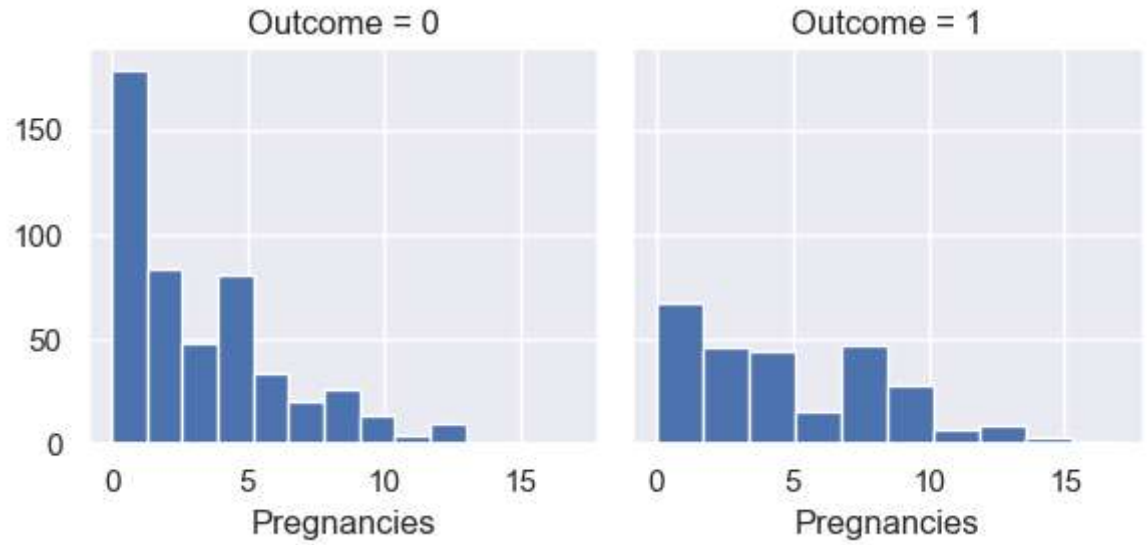
In [43]:
```
#Pregenecies
```

In [44]:
```
df1.columns
```

Out[44]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [45]:
```
Pregnancies=sns.FacetGrid(df1,col='Outcome')
Pregnancies.map(plt.hist,'Pregnancies')
```

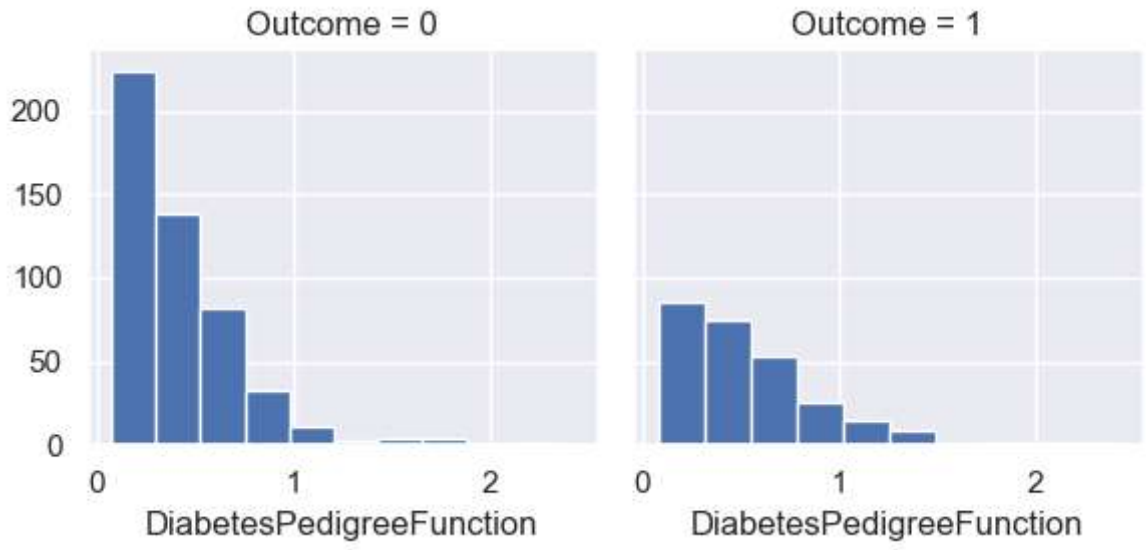
Out[45]:
```
<seaborn.axisgrid.FacetGrid at 0x16830d8fa10>
```



In [46]:
```
#diabetes Pedigree Function
```

In [47]:
```
DiabetesPedigreeFunction = sns.FacetGrid(df1, col='Outcome')
DiabetesPedigreeFunction.map(plt.hist, 'DiabetesPedigreeFunction')
```

Out[47]: <seaborn.axisgrid.FacetGrid at 0x16830bf9150>



In [48]:
```
#Nutritional Status Based on BMI
```

In [49]:
```
#Nutritional Status Source: World Health Organization.
```

In [50]:
```
Nutritional_Status= pd.Series([])
```

Nutritional_Status = []

for i in range(len(df1)): if df1['BMI'][i] == 0.0: Nutritional_Status.append("NA") elif df1['BMI'][i] < 18.5: Nutritional_Status.append("UnderWeight") elif df1['BMI'][i] < 25: Nutritional_Status.append("Normal") elif df1['BMI'][i] >= 25 and df1['BMI'][i] < 30: Nutritional_Status.append("Overweight") elif df1['BMI'][i] >= 30: Nutritional_Status.append("Obese") else: Nutritional_Status.append(df1['BMI'][i])

In [51]:
```
for i in range(len(df1)):
    if df1['BMI'][i] == 0.0:
        Nutritional_Status[i]="NA"
    elif df1['BMI'][i] < 18.5:
        Nutritional_Status[i]="Underweight"
    elif df1['BMI'][i] < 25:
        Nutritional_Status[i]="Normal"
    elif df1['BMI'][i] >= 25 and df1['BMI'][i] < 30:
        Nutritional_Status[i]="Overweight"
    elif df1['BMI'][i] >= 30:
        Nutritional_Status[i]="Obese"
    else:
        Nutritional_Status[i]= df1['BMI'][i]
```

In [52]:
```
df1.insert(6,'Nutritional_Status',Nutritional_Status )
```

In [53]:
```
df1.head()
```

Out[53]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Nutritional_Status	Diabete
0	6	148.0	72.0	35.0	125.0	33.6	Obese	
1	1	85.0	66.0	29.0	125.0	26.6	Overweight	
2	8	183.0	64.0	29.0	125.0	23.3	Normal	
3	1	89.0	66.0	23.0	94.0	28.1	Overweight	
4	0	137.0	40.0	35.0	168.0	43.1	Obese	

In [54]:
```
df1['Nutritional_Status'].value_counts()
```

Out[54]:
```
Obese          483
Overweight     179
Normal         102
Underweight      4
Name: Nutritional_Status, dtype: int64
```

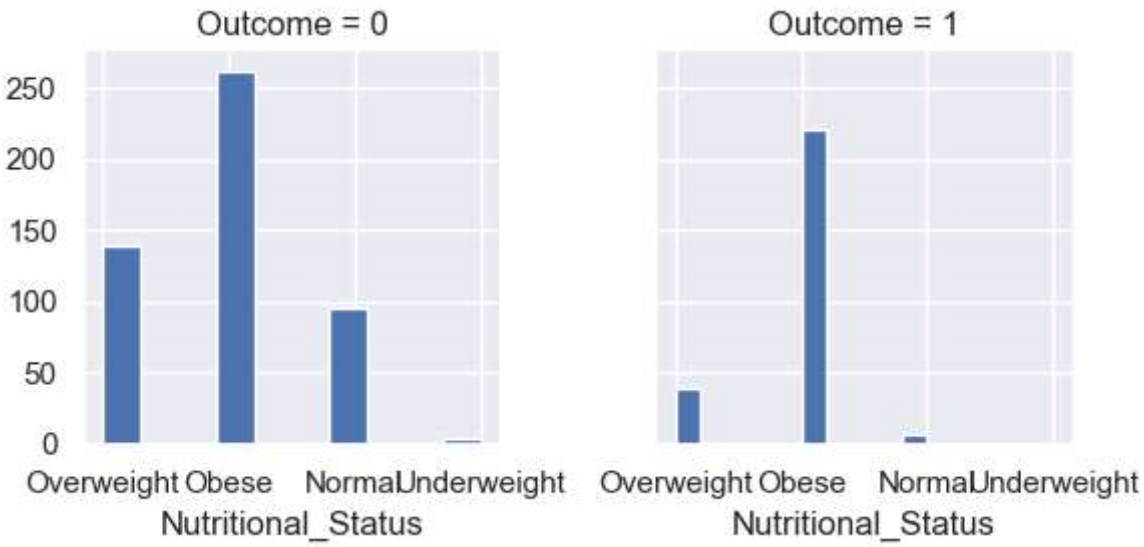
In [55]:
```
print(df1.columns)
df1.columns = df1.columns.str.strip()
df1['Nutritional_Status'].value_counts()
df1['Nutritional_Status'].value_counts()  # Using lowercase
df1['Nutritional_Status'].value_counts()  # Using uppercase
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'Nutritional_Status', 'DiabetesPedigreeFunction', 'Age',
       'Outcome'],
      dtype='object')
```

Out[55]:
```
Obese          483
Overweight     179
Normal         102
Underweight      4
Name: Nutritional_Status, dtype: int64
```

In [56]:
```
NutritionalStatus=sns.FacetGrid(df1,col='Outcome')
NutritionalStatus.map(plt.hist,'Nutritional_Status')
```

Out[56]:
```
<seaborn.axisgrid.FacetGrid at 0x168309c7a10>
```



In [57]:
```
OGTT = pd.Series([])
```

In [58]:
```
for i in range(len(df1)):
    if df1['Glucose'][i] == 0.0:
        OGTT [i]="NA"

    elif df1['Glucose'][i] <= 140:
        OGTT [i]="Normal"

    elif df1['Glucose'][i] > 140 and df1['Glucose'][i] <= 198:
        OGTT [i]="Impaired Glucose Tolerance"

    elif df1['Glucose'][i] > 198:
        OGTT [i]="Diabetic Level"
    else:
        OGTT [i]= df1['Glucose'][i]
```

In [59]:
```
 # Insert new column - Glucose Result
df1.insert(2, "Glucose Result", OGTT)
```

In [60]:
```
df1['Glucose Result'].value_counts()
```

Out[60]:
```
Normal                          576
Impaired Glucose Tolerance      191
Diabetic Level                    1
Name: Glucose Result, dtype: int64
```

In [61]:
```
Impaired_Glucose_Tolerance_Diabetic = ((df1['Glucose'] > 140) & (df1['Glucose']
Impaired_Glucose_Tolerance_Diabetic
```

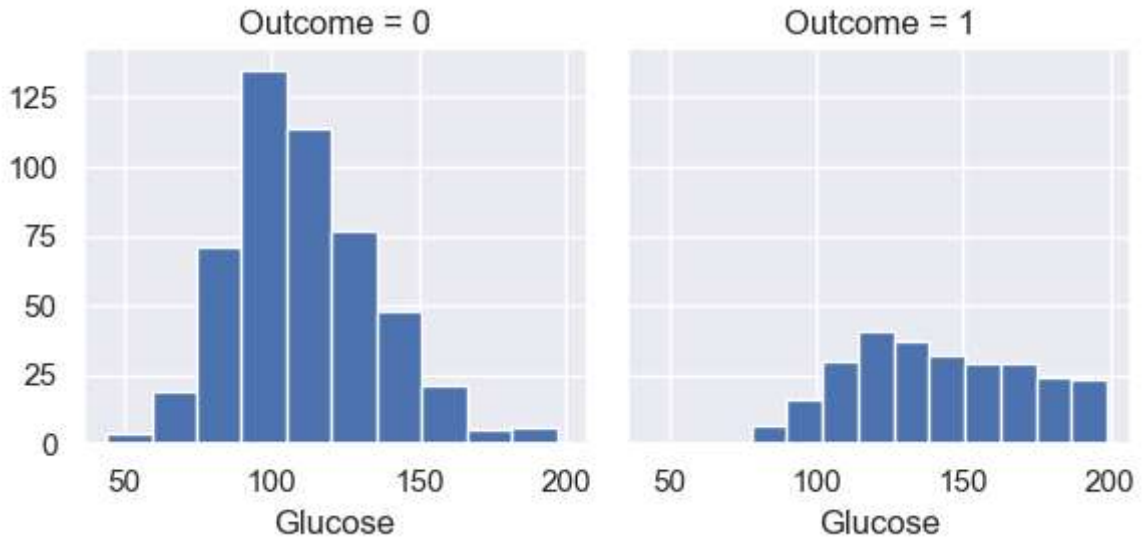
Out[61]: 131

In [62]:
```
Normal_Glucose_Diabetic = ((df1['Glucose'] != 0) & (df1['Glucose'] <= 140) & (d
Normal_Glucose_Diabetic
```

Out[62]: 136

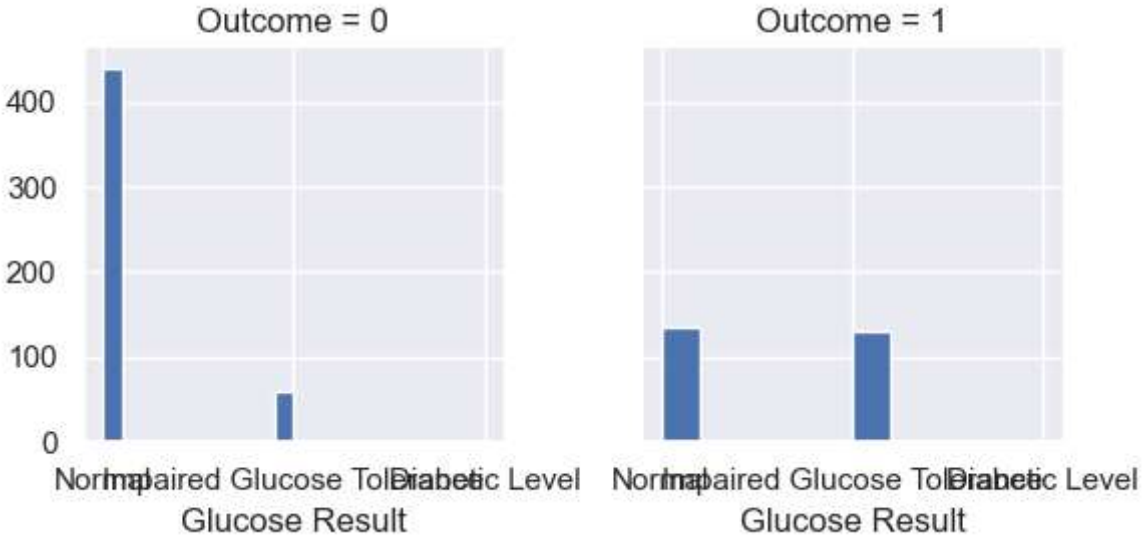
In [63]:
```
Glucose=sns.FacetGrid(df1,col='Outcome')
Glucose.map(plt.hist,'Glucose')
```

Out[63]: <seaborn.axisgrid.FacetGrid at 0x16830fffa10>



In [64]:
```
GlucoseResult=sns.FacetGrid(df1,col='Outcome')
GlucoseResult.map(plt.hist,'Glucose Result')
```

Out[64]: <seaborn.axisgrid.FacetGrid at 0x1683112cc50>



In [65]:
```
Percentile_skin_thickness = pd.Series([])
```

In [66]:
```
women_80_or_older = df1[(df1['Age'] >= 80) & (df1['Outcome'] == 1)].shape[0]
print(women_80_or_older)
```

```
0
```

In [67]:
```
df1['Age'].value_counts()
```

Out[67]:
```
22    72
21    63
25    48
24    46
23    38
28    35
26    33
27    32
29    29
31    24
41    22
30    21
37    19
42    18
33    17
38    16
36    16
32    16
45    15
34    14
46    13
43    13
40    13
39    12
35    10
50     8
51     8
52     8
44     8
58     7
47     6
54     6
49     5
48     5
57     5
53     5
60     5
66     4
63     4
62     4
55     4
67     3
56     3
59     3
65     3
69     2
61     2
72     1
81     1
64     1
70     1
68     1
Name: Age, dtype: int64
```

In [68]:
```
for i in range(len(df1)):
    if df1["Age"][i] >= 20.0 and df1["Age"][i] <= 79.0:
        if df1["SkinThickness"][i] == 0.0:
            Percentile_skin_thickness[i]=" 0 NA"
        elif df1["SkinThickness"][i] < 11.9:
            Percentile_skin_thickness[i]="1 <P5th"

        elif df1["SkinThickness"][i] == 11.9:
            Percentile_skin_thickness[i]="2 P5th"

        elif df1["SkinThickness"][i] > 11.9 and df1["SkinThickness"][i] < 14.0:
            Percentile_skin_thickness[i]="3 P5th - P10th"

        elif df1["SkinThickness"][i] == 14.0:
            Percentile_skin_thickness[i]="4 P10th"

        elif df1["SkinThickness"][i] > 14.0 and df1["SkinThickness"][i] < 15.8:
            Percentile_skin_thickness[i]="5 P10th - P15th"

        elif df1["SkinThickness"][i] == 15.8:
            Percentile_skin_thickness[i]="6 P15th"

        elif df1["SkinThickness"][i] > 15.8 and df1["SkinThickness"][i] < 18.0:
            Percentile_skin_thickness[i]="7 P15th - P25th"

        elif df1["SkinThickness"][i] == 18.0:
            Percentile_skin_thickness[i]="8 P25th"

        elif df1["SkinThickness"][i] > 18.0 and df1["SkinThickness"][i] < 23.5:
            Percentile_skin_thickness[i]="9 P25th - P50th"

        elif df1["SkinThickness"][i] == 23.5:
            Percentile_skin_thickness[i]="10 P50th"

        elif df1["SkinThickness"][i] > 23.5 and df1["SkinThickness"][i] < 29.0:
            Percentile_skin_thickness[i]="11 P50th - P75th"

        elif df1["SkinThickness"][i] == 29.0:
            Percentile_skin_thickness[i]="12 P75th"

        elif df1["SkinThickness"][i] > 29.0 and df1["SkinThickness"][i] < 31.9:
            Percentile_skin_thickness[i]="13 P75th - P85th"

        elif df1["SkinThickness"][i] == 31.9:
            Percentile_skin_thickness[i]="14 P85th"

        elif df1["SkinThickness"][i] > 31.9 and df1["SkinThickness"][i] < 33.7:
            Percentile_skin_thickness[i]="15 P85th - P90th"

        elif df1["SkinThickness"][i] == 33.7:
            Percentile_skin_thickness[i]="16 P90th"

        elif df1["SkinThickness"][i] > 33.7 and df1["SkinThickness"][i] < 35.9:
            Percentile_skin_thickness[i]="17 P90th - P95th"

        elif df1["SkinThickness"][i] == 35.9:
            Percentile_skin_thickness[i]="18 P95th"
```

```
        elif df1["SkinThickness"][i] > 35.9:
            Percentile_skin_thickness[i]="19 >P95th"

        elif df1["Age"][i] >= 80.0: #Only 1 woman is 81 years old
            if df1["SkinThickness"][i] > 31.7:
                Percentile_skin_thickness[i]="20 >P95th"
```

In [69]:
```
df1.insert(4, "Percentile skin thickness", Percentile_skin_thickness)
```

In [70]:
```
df1.head(5)
```

Out[70]:

	Pregnancies	Glucose	Glucose Result	BloodPressure	Percentile skin thickness	SkinThickness	Insulin	BMI	Nutr
0	6	148.0	Impaired Glucose Tolerance	72.0	17 P90th - P95th	35.0	125.0	33.6	
1	1	85.0	Normal	66.0	12 P75th	29.0	125.0	26.6	
2	8	183.0	Impaired Glucose Tolerance	64.0	12 P75th	29.0	125.0	23.3	
3	1	89.0	Normal	66.0	9 P25th - P50th	23.0	94.0	28.1	
4	0	137.0	Normal	40.0	17 P90th - P95th	35.0	168.0	43.1	

In [71]:
```
# Check number of women x Percentile of skin thickness
df1['Percentile skin thickness'].value_counts()
```

Out[71]:
```
12 P75th            244
19 >P95th           145
11 P50th - P75th     87
9 P25th - P50th      79
15 P85th - P90th     50
13 P75th - P85th     46
17 P90th - P95th     23
8 P25th              20
7 P15th - P25th      20
3 P5th - P10th       18
1 <P5th              15
5 P10th - P15th      14
4 P10th               6
Name: Percentile skin thickness, dtype: int64
```

In [72]:
```
diabetic_malnourished_st = ((df1['SkinThickness'] < 15.8) & (df1['Outcome'] ==
diabetic_malnourished_st
```

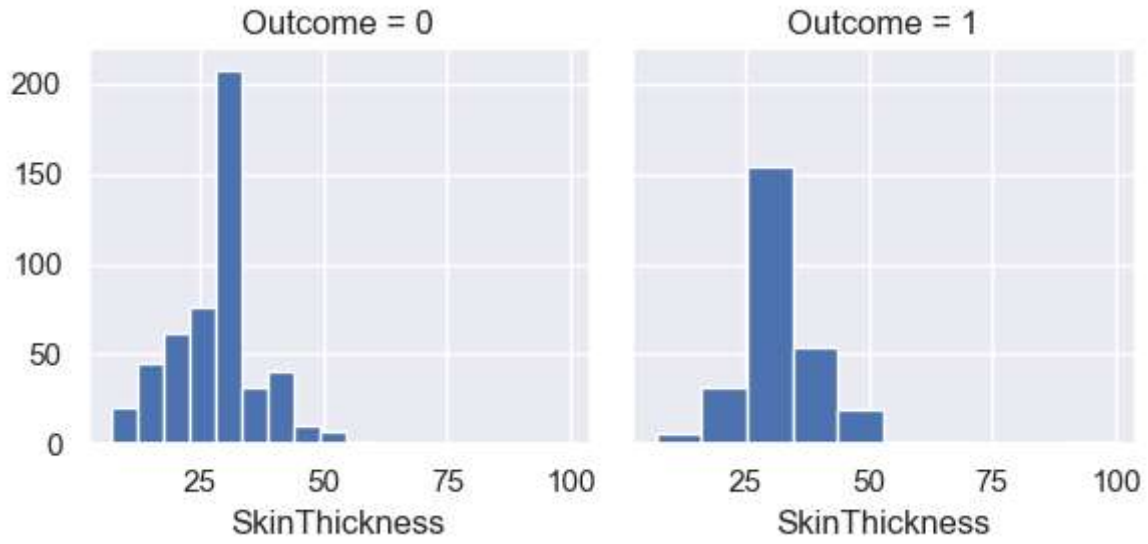
Out[72]:
```
6
```

In [73]:
```
df.mean()
```

Out[73]:
```
Pregnancies                  3.845052
Glucose                    120.894531
BloodPressure               69.105469
SkinThickness               20.536458
Insulin                     79.799479
BMI                         31.992578
DiabetesPedigreeFunction     0.471876
Age                         33.240885
Outcome                      0.348958
dtype: float64
```

In [74]:
```
SkinThickness=sns.FacetGrid(df1,col='Outcome')
SkinThickness.map(plt.hist,'SkinThickness')
```

Out[74]:
```
<seaborn.axisgrid.FacetGrid at 0x16830b0fa10>
```



In [75]:
```
#Blood Pressure
```

In [76]:
```
df1['BloodPressure'].mean()
```

Out[76]:
```
72.40518417462482
```

In [77]:
```
df1['BloodPressure'].min()
```

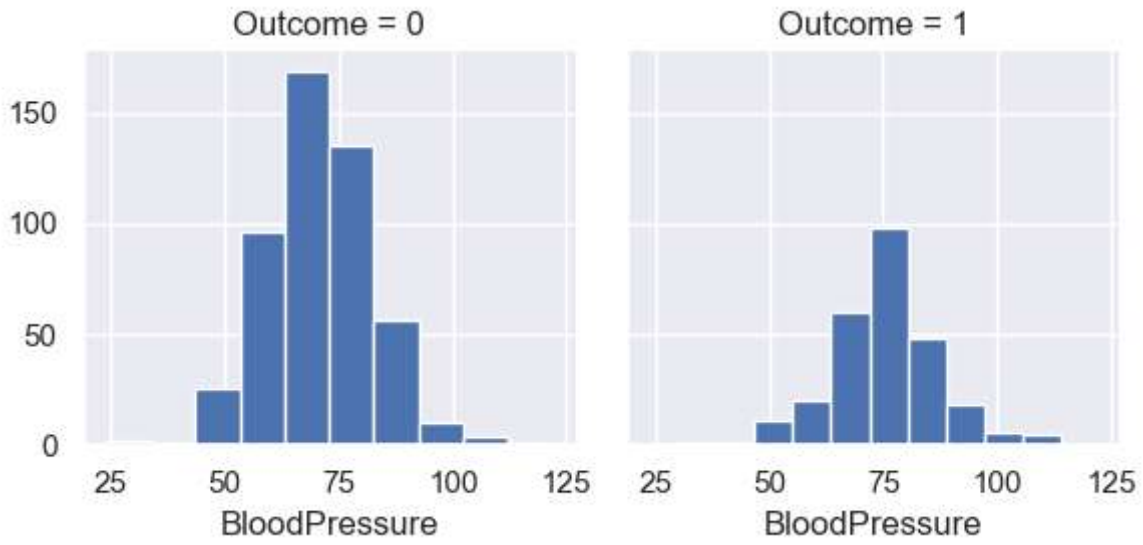
Out[77]:
```
24.0
```

In [78]:
```
df1['BloodPressure'].max()
```

Out[78]:
```
122.0
```

In [79]:
```
BloodPressure=sns.FacetGrid(df1,col='Outcome')
BloodPressure.map(plt.hist,'BloodPressure')
```

Out[79]: <seaborn.axisgrid.FacetGrid at 0x16830abfa10>



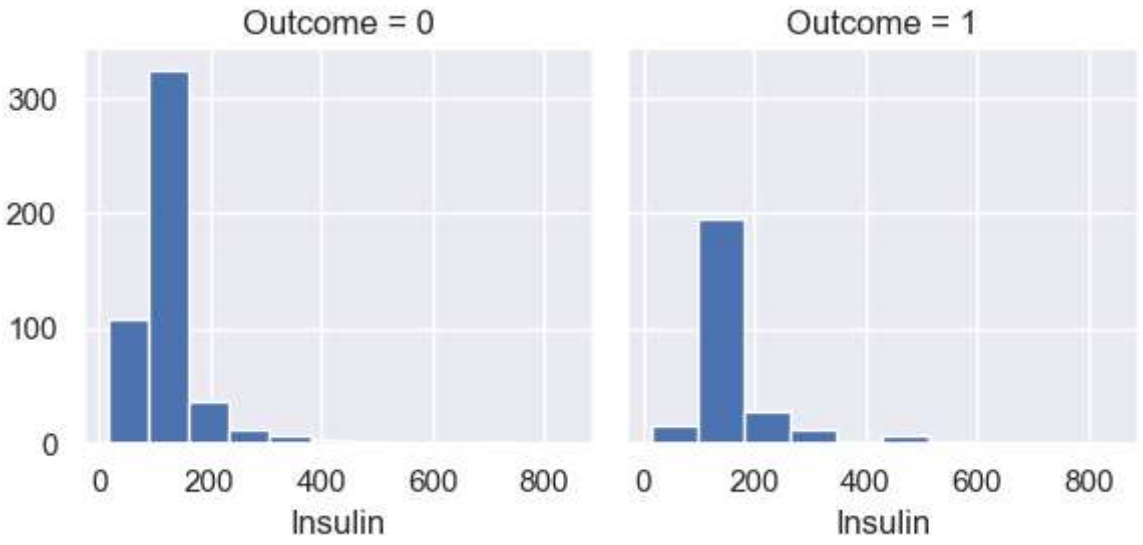
In [80]:
```
#Insulin
```

In [81]:
```
df1['Insulin'].mean()
```

Out[81]: 140.671875

In [82]:
```
Insulin=sns.FacetGrid(df1,col='Outcome')
Insulin.map(plt.hist,'Insulin')
```

Out[82]: <seaborn.axisgrid.FacetGrid at 0x16831217a10>



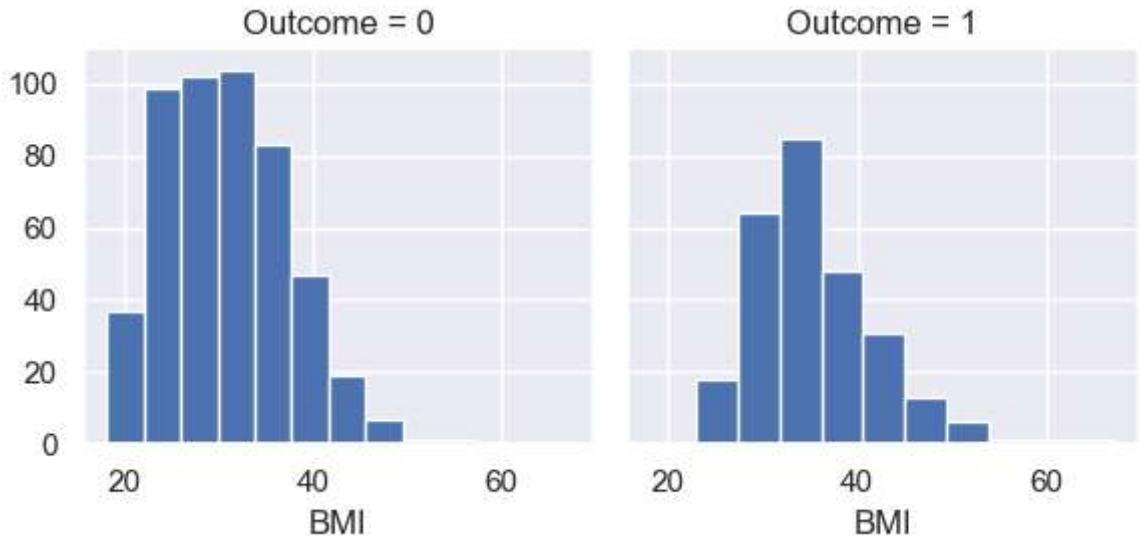
In [83]:
```
#BMI
```

In [84]:
```
df1['BMI'].mean()
```

Out[84]:
```
32.45520833333333
```

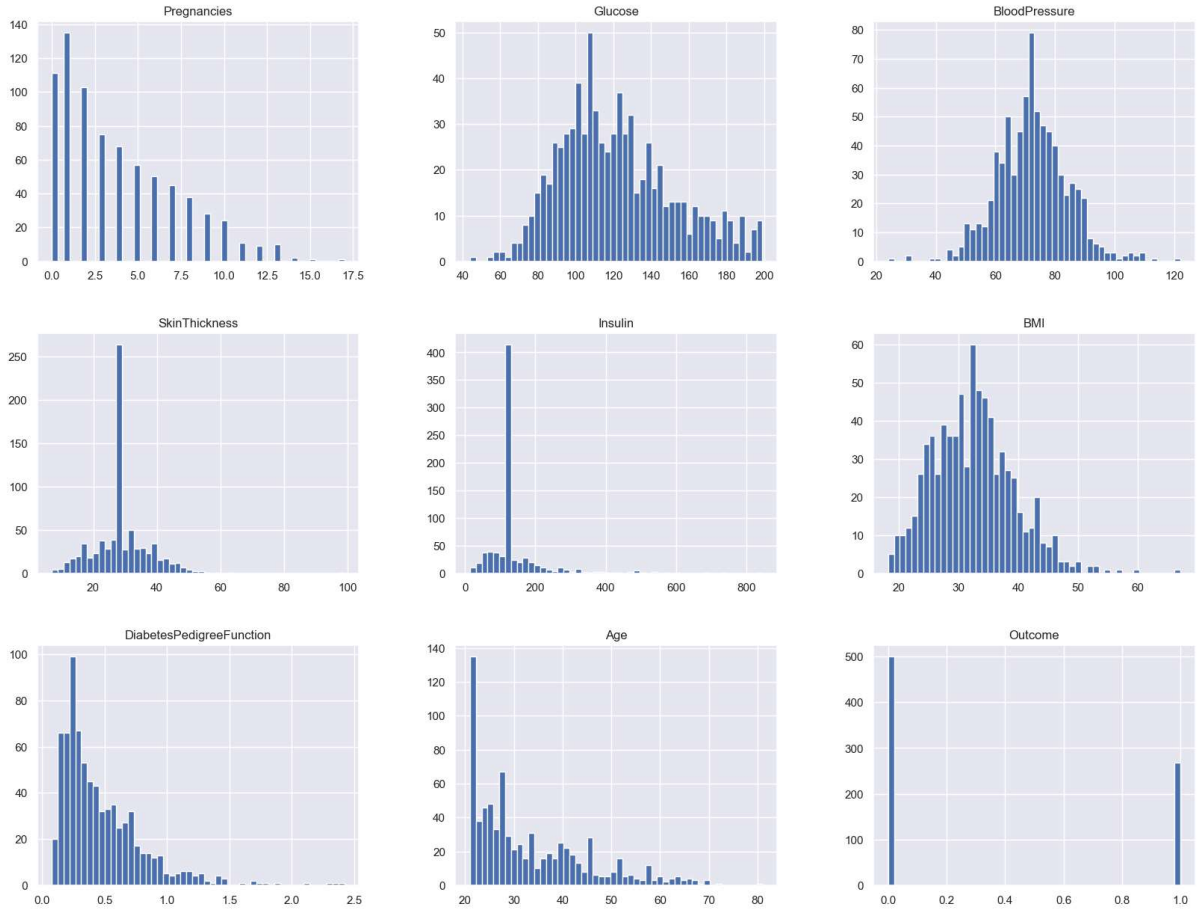
In [85]:
```
BMI=sns.FacetGrid(df1,col='Outcome')
BMI.map(plt.hist,'BMI')
```

Out[85]:
```
<seaborn.axisgrid.FacetGrid at 0x168309ed490>
```



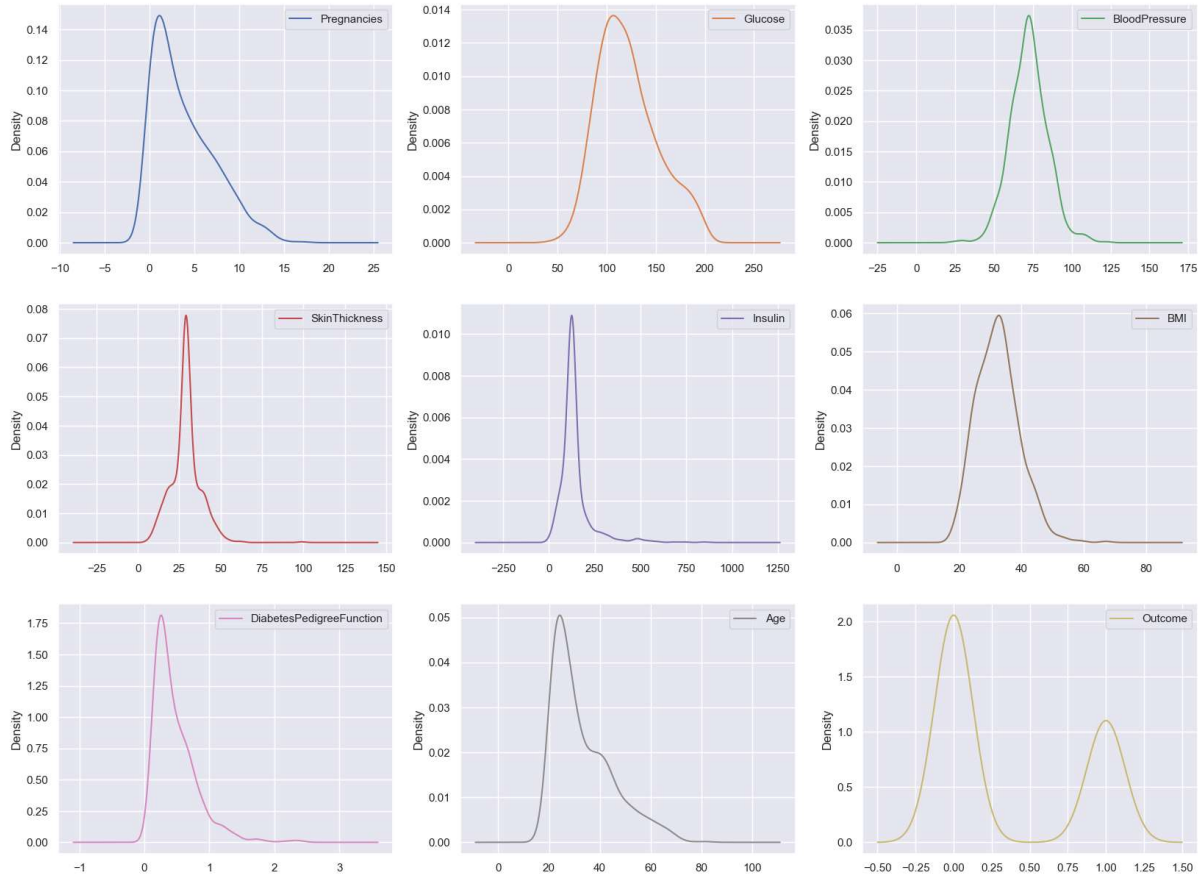
In [86]:
```
#Data Visualization
```

In [87]:
```
 # Histogram
df1.hist(bins=50, figsize=(20, 15))
plt.show()
```



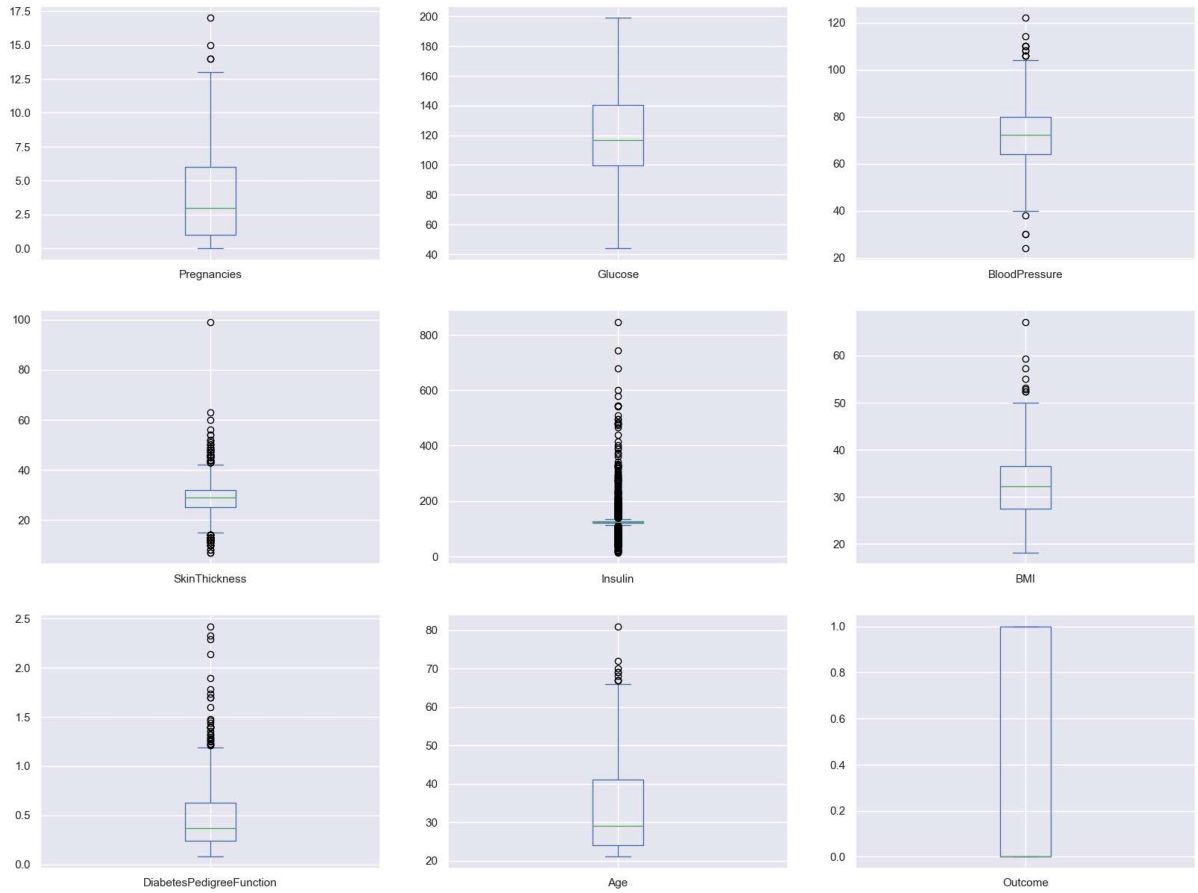
In [88]:
```
#Distribution Plot
```

In [89]:
```
df1.plot(kind='density', subplots=True, layout=(3,3), figsize=(20, 15), sharex=
plt.show()
```



In [90]:
```
df1.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False, f
```

Out[90]:
```
Pregnancies                 Axes(0.125,0.653529;0.227941x0.226471)
Glucose                  Axes(0.398529,0.653529;0.227941x0.226471)
BloodPressure            Axes(0.672059,0.653529;0.227941x0.226471)
SkinThickness               Axes(0.125,0.381765;0.227941x0.226471)
Insulin                  Axes(0.398529,0.381765;0.227941x0.226471)
BMI                      Axes(0.672059,0.381765;0.227941x0.226471)
DiabetesPedigreeFunction        Axes(0.125,0.11;0.227941x0.226471)
Age                      Axes(0.398529,0.11;0.227941x0.226471)
Outcome                  Axes(0.672059,0.11;0.227941x0.226471)
dtype: object
```



In [91]:
```
#Skew of attributes distributions
skew=df1.skew(axis = 1)
```

In [92]:
```
skew
```

Out[92]:
```
0      0.921937
1      1.015506
2      1.468154
3      0.711041
4      1.339072
         ...
763    1.228541
764    0.904054
765    0.907577
766    0.915431
767    0.935570
Length: 768, dtype: float64
```

In [93]:
```
#Pairplot
```

In [94]:
```
sns.pairplot(df1, hue='Outcome')
```

Out[94]:
```
<seaborn.axisgrid.PairGrid at 0x16833f11d90>
```

In [95]:
```
#Correlation between all the features
```

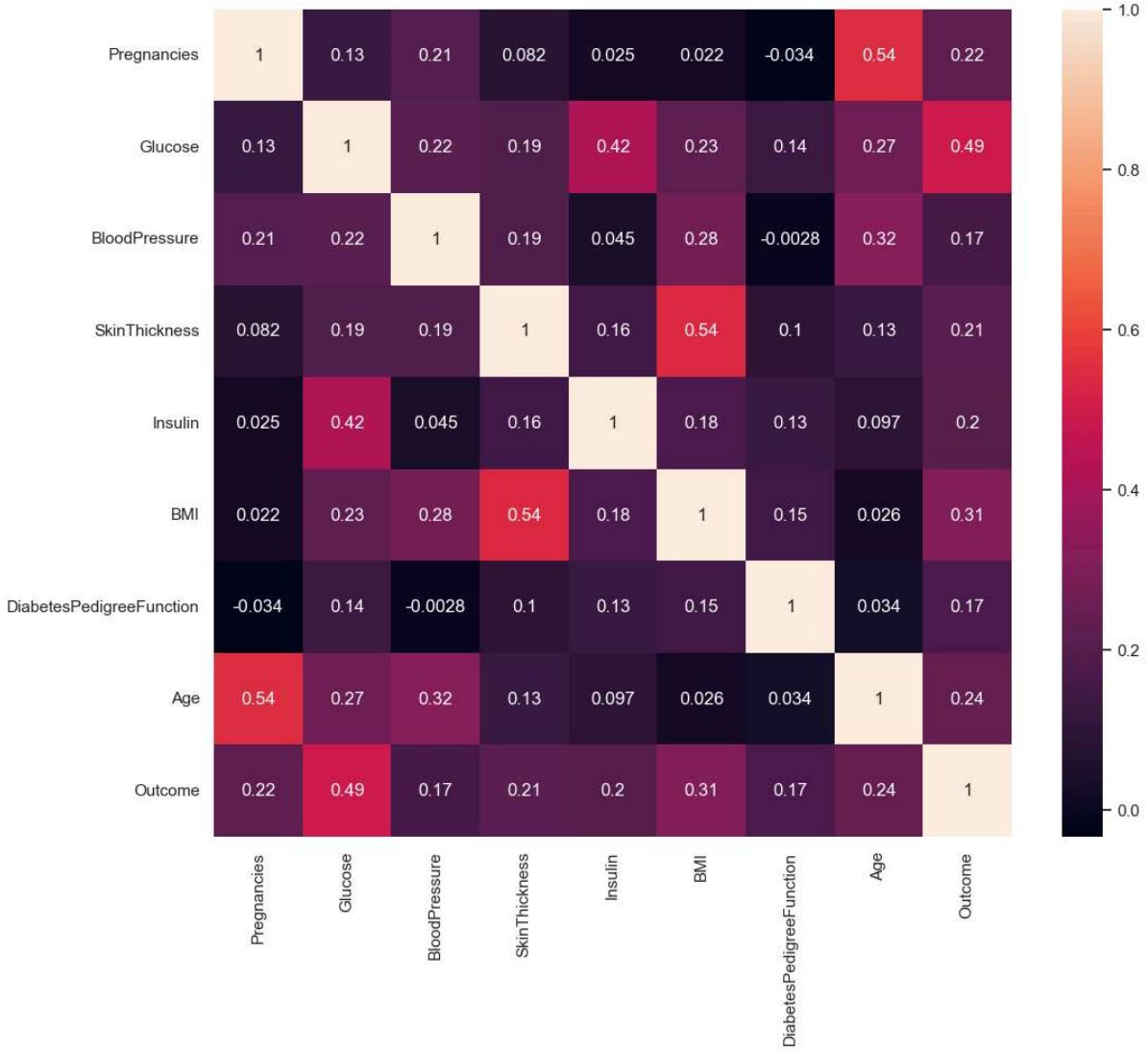
In [96]:
```
corr_matrix = df1.corr(method='pearson')
corr_matrix
```

Out[96]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	B
Pregnancies	1.000000	0.127911	0.208522	0.081770	0.025047	0.0215
Glucose	0.127911	1.000000	0.218367	0.192686	0.419064	0.2311
BloodPressure	0.208522	0.218367	1.000000	0.191853	0.045087	0.2811
SkinThickness	0.081770	0.192686	0.191853	1.000000	0.155610	0.5432
Insulin	0.025047	0.419064	0.045087	0.155610	1.000000	0.1802
BMI	0.021559	0.231128	0.281199	0.543205	0.180241	1.0000
DiabetesPedigreeFunction	-0.033523	0.137060	-0.002763	0.102188	0.126503	0.1534
Age	0.544341	0.266534	0.324595	0.126107	0.097101	0.0255
Outcome	0.221898	0.492928	0.166074	0.214873	0.203790	0.3120

In [97]:
```
plt.figure(figsize=(12,10))
sns.heatmap(corr_matrix, annot = True)
```

Out[97]: <Axes: >



In [98]:
```
#Separating the Features and Target
```

In [99]:
```
df1.columns
```

Out[99]:
```
Index(['Pregnancies', 'Glucose', 'Glucose Result', 'BloodPressure',
       'Percentile skin thickness', 'SkinThickness', 'Insulin', 'BMI',
       'Nutritional_Status', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [100]:
```
df1.head()
```

Out[100]:

	Pregnancies	Glucose	Glucose Result	BloodPressure	Percentile skin thickness	SkinThickness	Insulin	BMI	Nutr
0	6	148.0	Impaired Glucose Tolerance	72.0	17 P90th - P95th	35.0	125.0	33.6	
1	1	85.0	Normal	66.0	12 P75th	29.0	125.0	26.6	
2	8	183.0	Impaired Glucose Tolerance	64.0	12 P75th	29.0	125.0	23.3	
3	1	89.0	Normal	66.0	9 P25th - P50th	23.0	94.0	28.1	
4	0	137.0	Normal	40.0	17 P90th - P95th	35.0	168.0	43.1	

In [101]:
```
X = df1.drop(columns=['Outcome', 'Glucose Result', 'Percentile skin thickness',
Y = df1['Outcome']
```

In [102]:
```
X.head()
```

Out[102]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.0	125.0	33.6	0.627
1	1	85.0	66.0	29.0	125.0	26.6	0.351
2	8	183.0	64.0	29.0	125.0	23.3	0.672
3	1	89.0	66.0	23.0	94.0	28.1	0.167
4	0	137.0	40.0	35.0	168.0	43.1	2.288

In [103]:
```
Y.head()
```

Out[103]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [104]:
```
#Data Preprocessing
# Standardization
```

In [105]:
```
scaler = StandardScaler()
```

In [106]:
```
scaler.fit(X)
```

Out[106]:
```
StandardScaler()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [107]:
```
StandardScaler()
```

Out[107]:
```
StandardScaler()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [108]:
```
standardized_data = scaler.transform(X)
```

In [109]:
```
print(standardized_data)
```

```
[[ 0.63994726  0.86510807 -0.03351824 ...  0.16661938  0.46849198
   1.4259954 ]
 [-0.84488505 -1.20616153 -0.52985903 ... -0.85219976 -0.36506078
  -0.19067191]
 [ 1.23388019  2.0158134  -0.69530596 ... -1.33250021  0.60439732
  -0.10558415]
 ...
 [ 0.3429808  -0.0225789  -0.03351824 ... -0.910418   -0.68519336
  -0.27575966]
 [-0.84488505  0.14180757 -1.02619983 ... -0.34279019 -0.37110101
   1.17073215]
 [-0.84488505 -0.94314317 -0.19896517 ... -0.29912651 -0.47378505
  -0.87137393]]
```

In [110]:
```
X_sc = standardized_data
```

In [111]:
```
print(X_sc)
```

```
[[ 0.63994726  0.86510807 -0.03351824 ...  0.16661938  0.46849198
   1.4259954 ]
 [-0.84488505 -1.20616153 -0.52985903 ... -0.85219976 -0.36506078
  -0.19067191]
 [ 1.23388019  2.0158134  -0.69530596 ... -1.33250021  0.60439732
  -0.10558415]
 ...
 [ 0.3429808  -0.0225789  -0.03351824 ... -0.910418   -0.68519336
  -0.27575966]
 [-0.84488505  0.14180757 -1.02619983 ... -0.34279019 -0.37110101
   1.17073215]
 [-0.84488505 -0.94314317 -0.19896517 ... -0.29912651 -0.47378505
  -0.87137393]]
```

In [112]:
```
#Train Test Split
```

In [113]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, strati
```

In [114]:
```
print(X_sc.shape, X_train.shape, X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

In [115]:
```
#Training the Model
#K Neighbor Classifier
```

In [116]:
```
knn = nei.KNeighborsClassifier(n_neighbors=5)
```

In [117]:
```
#training the support vector Machine Classifier
knn.fit(X_train, Y_train)
```

Out[117]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [118]:
```
#Accuracy Score
```

In [119]:
```
# accuracy score on the training data
X_train_prediction = knn.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [120]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.7980456026058632
```

In [121]:
```
# accuracy score on the test data
X_test_prediction = knn.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [122]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7207792207792207
```

In [123]:
```
 #Evaluation:
```

In [124]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[84 16]
 [27 27]]
              precision    recall  f1-score   support

           0       0.76      0.84      0.80       100
           1       0.63      0.50      0.56        54

    accuracy                           0.72       154
   macro avg       0.69      0.67      0.68       154
weighted avg       0.71      0.72      0.71       154
```

In [125]:
```
#Support Vector Machine
```

In [126]:
```
classifier = svm.SVC(kernel='linear')
```

In [127]:
```
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

Out[127]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [128]:
```
#Model Evaluation
#Accuracy Score
```

In [129]:
```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [130]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  0.7752442996742671
```

In [131]:
```
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [132]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7727272727272727
```

In [133]:
```
 #Evaluation
```

In [134]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[91  9]
 [26 28]]
              precision    recall  f1-score   support

           0       0.78      0.91      0.84       100
           1       0.76      0.52      0.62        54

    accuracy                           0.77       154
   macro avg       0.77      0.71      0.73       154
weighted avg       0.77      0.77      0.76       154
```

In [135]:
```
#Decision Tree
```

In [136]:
```
dtree = DecisionTreeClassifier()
```

In [137]:
```
dtree.fit(X_train, Y_train)
```

Out[137]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [138]:
```
# accuracy score on the training data
X_train_prediction = dtree.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [139]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  1.0
```

In [140]:
```
# accuracy score on the test data
X_test_prediction = dtree.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [141]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.6623376623376623
```

In [142]:
```
#Evaluation
```

In [143]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[78 22]
 [30 24]]
              precision    recall  f1-score   support

           0       0.72      0.78      0.75       100
           1       0.52      0.44      0.48        54

    accuracy                           0.66       154
   macro avg       0.62      0.61      0.61       154
weighted avg       0.65      0.66      0.66       154
```

In [144]:
```
#RandomForest
```

In [145]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

In [146]:
```
rfc.fit(X_train, Y_train)
```

Out[146]:
```
RandomForestClassifier(n_estimators=200)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [147]:
```
# accuracy score on the training data
X_train_prediction = rfc.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [148]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  1.0
```

In [149]:
```
# accuracy score on the test data
X_test_prediction = rfc.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [150]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7467532467532467
```

In [151]:
```
#Evaluation
```

In [152]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[85 15]
 [24 30]]
              precision    recall  f1-score   support

           0       0.78      0.85      0.81       100
           1       0.67      0.56      0.61        54

    accuracy                           0.75       154
   macro avg       0.72      0.70      0.71       154
weighted avg       0.74      0.75      0.74       154
```

In [153]:
```
## XgBoost
```

In [154]:
```
xgb_model = XGBClassifier(gamma=0)
```

In [155]:
```
xgb_model.fit(X_train, Y_train)
```

Out[155]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0, 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,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [156]:
```
# accuracy score on the training data
X_train_prediction = xgb_model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [157]:
```
print('Accuracy score of the training data : ', training_data_accuracy)
```

```
Accuracy score of the training data :  1.0
```

In [158]:
```
# accuracy score on the test data
X_test_prediction = xgb_model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [159]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7532467532467533
```

In [171]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[81 19]
 [19 35]]
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       100
           1       0.65      0.65      0.65        54

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```

In [172]:
```
#Cross Validation
#Recursive feature elimination
```

In [173]:
```
# Define KFold
kf = KFold(n_splits=10, shuffle=False, random_state=None)
```

In [174]:
```
skf = StratifiedKFold(n_splits=10, random_state=None)
```

In [175]:
```
from sklearn.svm import SVC

classifier = SVC(kernel='linear')
```

In [178]:
```
print(dir(classifier))
```

```
['C', '__abstractmethods__', '__annotations__', '__class__', '__delattr__',
'__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getatt
ribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass
__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__
reduce_ex__', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__skl
earn_clone__', '__str__', '__subclasshook__', '__weakref__', '_abc_impl', '_b
uild_request_for_signature', '_check_feature_names', '_check_n_features', '_c
heck_proba', '_class_weight', '_compute_kernel', '_decision_function', '_dens
e_decision_function', '_dense_fit', '_dense_predict', '_dense_predict_proba',
'_dual_coef_', '_estimator_type', '_gamma', '_get_coef', '_get_default_reques
ts', '_get_metadata_request', '_get_param_names', '_get_tags', '_impl', '_int
ercept_', '_more_tags', '_n_support', '_num_iter', '_parameter_constraints',
'_probA', '_probB', '_repr_html_', '_repr_html_inner', '_repr_mimebundle_',
'_sparse', '_sparse_decision_function', '_sparse_fit', '_sparse_kernels', '_s
parse_predict', '_sparse_predict_proba', '_validate_data', '_validate_for_pre
dict', '_validate_params', '_validate_targets', '_warn_from_fit_status', 'bre
ak_ties', 'cache_size', 'class_weight', 'class_weight_', 'classes_', 'coef0',
'coef_', 'decision_function', 'decision_function_shape', 'degree', 'dual_coef
_', 'epsilon', 'feature_names_in_', 'fit', 'fit_status_', 'gamma', 'get_metad
ata_routing', 'get_params', 'intercept_', 'kernel', 'max_iter', 'n_features_i
n_', 'n_iter_', 'n_support_', 'nu', 'predict', 'predict_log_proba', 'predict_
proba', 'probA_', 'probB_', 'probability', 'random_state', 'score', 'set_fit_
request', 'set_params', 'set_score_request', 'shape_fit_', 'shrinking', 'supp
ort_', 'support_vectors_', 'tol', 'unused_param', 'verbose']
```

In [180]:
```
classifier.fit(X_train, Y_train)
```

Out[180]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [165]:
```
 rfecv = RFECV (estimator=classifier,step=1, cv=skf, scoring='accuracy')
```

In [166]:
```
rfecv.fit(X,Y)
```

Out[166]:
```
RFECV(cv=StratifiedKFold(n_splits=10, random_state=None, shuffle=False),
      estimator=SVC(kernel='linear'), scoring='accuracy')
```

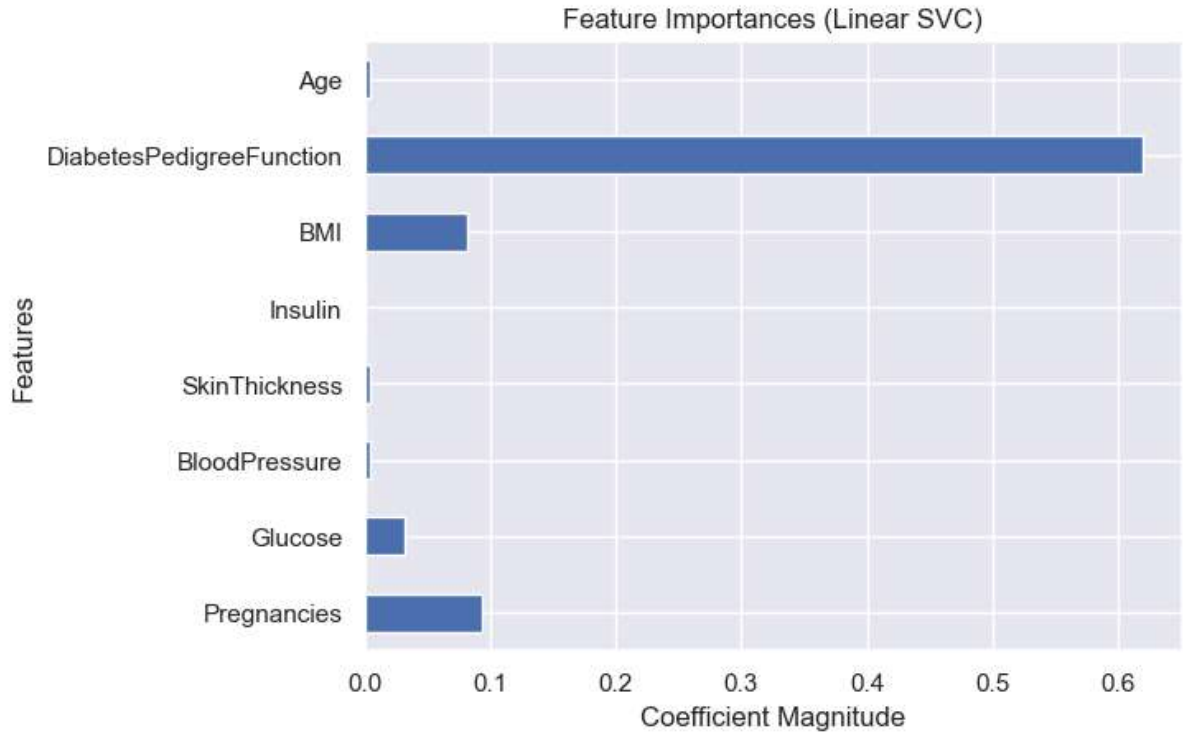
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [167]:
```
#Feature Importance
```

In [179]:
```
# Extracting feature coefficients (only applicable for linear kernel)
coefficients = classifier.coef_[0]
# Creating a pandas Series with feature importances
feature_importance = pd.Series(abs(coefficients), index=X_train.columns)
print(feature_importance)
```

```
Pregnancies                 0.093230
Glucose                     0.031556
BloodPressure               0.004559
SkinThickness               0.003791
Insulin                     0.000926
BMI                         0.081156
DiabetesPedigreeFunction    0.619352
Age                         0.005071
dtype: float64
```

In [182]:
```
# Extracting feature coefficients (only applicable for linear kernel)
coefficients = classifier.coef_[0]
# Creating a pandas Series with feature importances
feature_importance = pd.Series(abs(coefficients), index=X.columns)
# Plotting feature importances
feature_importance.plot(kind='barh')
plt.title("Feature Importances (Linear SVC)")
plt.xlabel("Coefficient Magnitude")
plt.ylabel("Features")
plt.show()
```



In [183]:
```
feature_names = X.columns[:10]
feature_names
```

Out[183]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')
```

In [184]:
```
X1 = X[feature_names]
```

In [185]:
```
new_features = list(filter(lambda x: x[1],zip(feature_names, rfecv.support_)))
new_features
```

Out[185]:
```
[('Pregnancies', True),
 ('Glucose', True),
 ('SkinThickness', True),
 ('BMI', True),
 ('DiabetesPedigreeFunction', True)]
```

In [186]:
```
X_new = df1[['Pregnancies','Glucose', 'BloodPressure','SkinThickness']]
```

In [187]:
```
scaler.fit(X_new)
```

Out[187]:
```
StandardScaler()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [188]:
```
standardized_data = scaler.transform(X_new)
```

In [189]:
```
Xnew_sc = standardized_data
```

In [190]:
```
Xnew_sc
```

Out[190]:
```
array([[ 0.63994726,  0.86510807, -0.03351824,  0.67064253],
       [-0.84488505, -1.20616153, -0.52985903, -0.01230129],
       [ 1.23388019,  2.0158134 , -0.69530596, -0.01230129],
       ...,
       [ 0.3429808 , -0.0225789 , -0.03351824, -0.69524511],
       [-0.84488505,  0.14180757, -1.02619983, -0.01230129],
       [-0.84488505, -0.94314317, -0.19896517,  0.21534665]])
```

In [192]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(Xnew_sc, Y, test_size=0.2,
```

In [193]:
```
print(Xnew_sc.shape, X_train.shape, X_test.shape)
```

```
(768, 4) (614, 4) (154, 4)
```

In [194]:
```
 classifier = svm.SVC(kernel='linear',probability=True)
```

In [195]:
```
 #training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
```

Out[195]:
```
SVC(kernel='linear', probability=True)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [196]:
```
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [197]:
```
print('Accuracy score of the test data : ', test_data_accuracy)
```

```
Accuracy score of the test data :  0.7532467532467533
```

In [198]:
```
#Evaluation
```

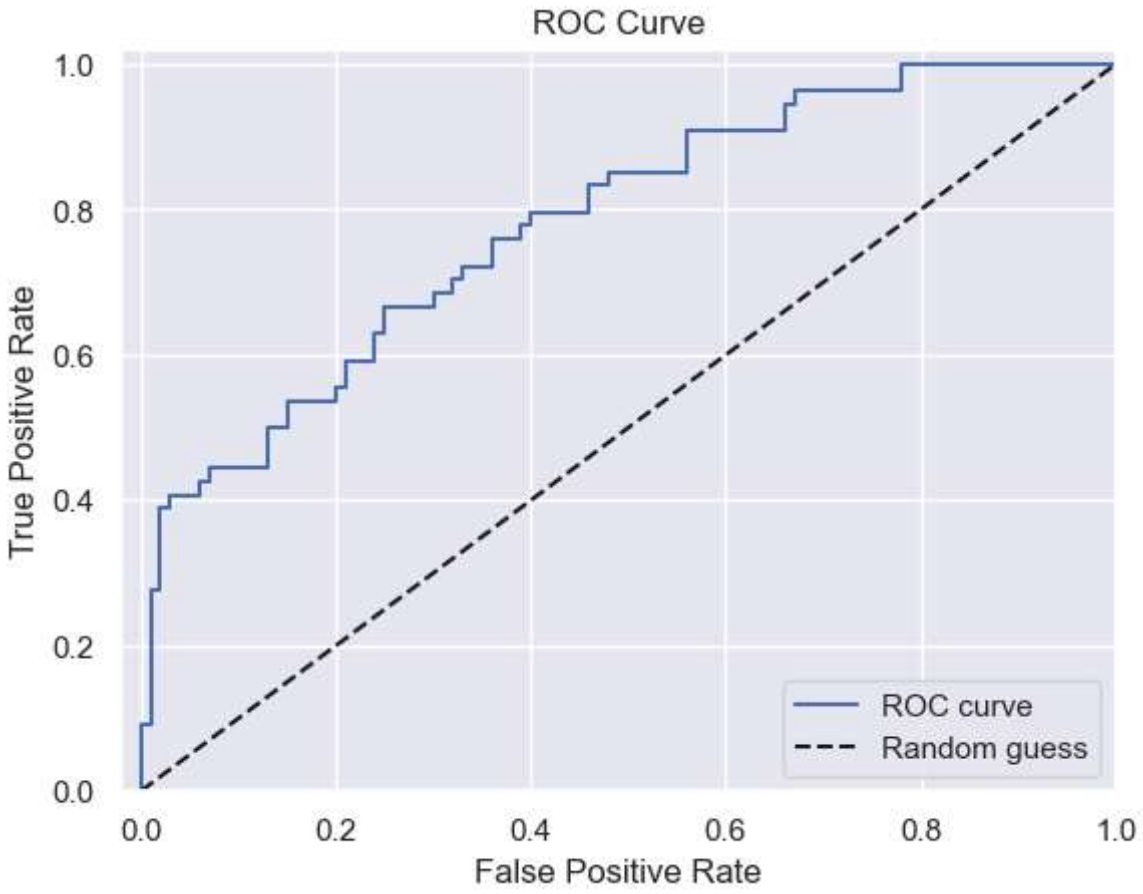
In [199]:
```
print(confusion_matrix(Y_test,X_test_prediction ))
print(classification_report(Y_test, X_test_prediction))
```

```
[[81 19]
 [19 35]]
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       100
           1       0.65      0.65      0.65        54

    accuracy                           0.75       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.75      0.75      0.75       154
```

In [200]:
```
 #ROC Curve
```

In [201]:
```
# Obtain predicted probabilities for the positive class (class 1)
out_pred_prob = classifier.predict_proba(X_test)[:, 1]
```

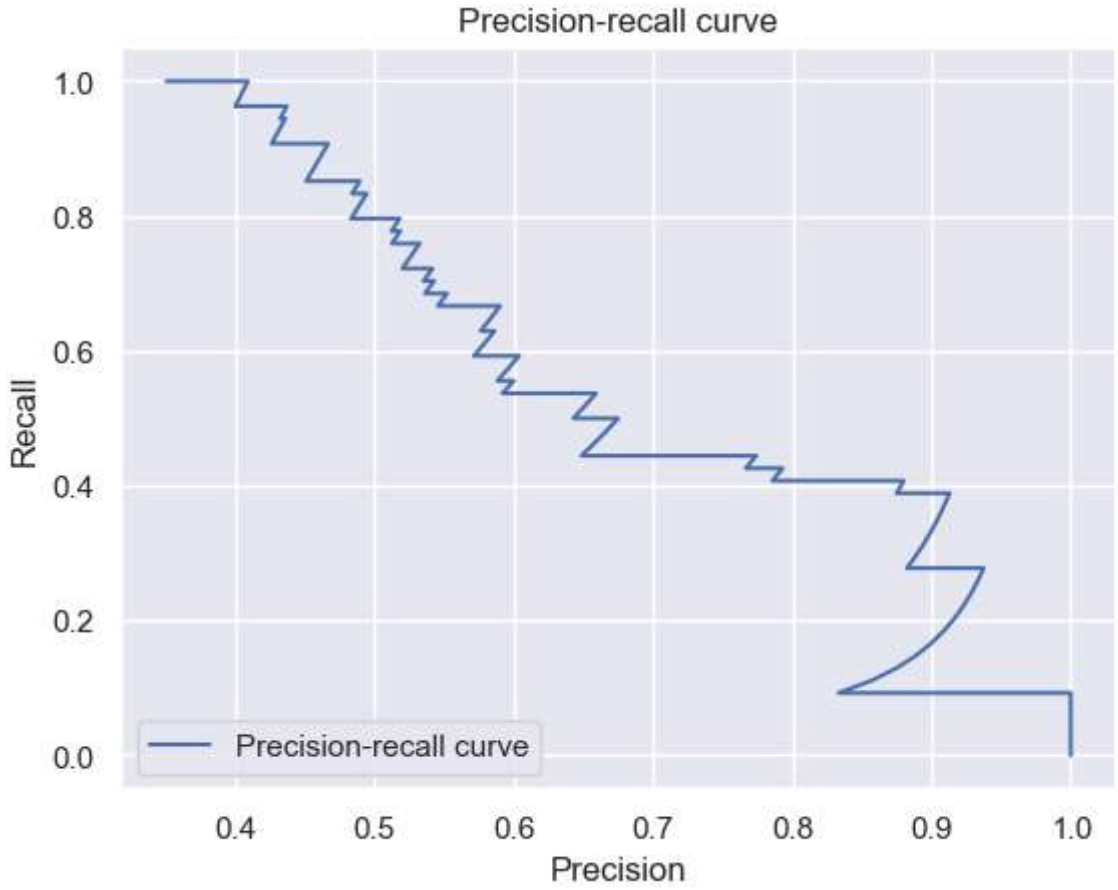
In [202]:
```
# Calculate false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(Y_test, out_pred_prob)
# Plot ROC curve
plt.plot(fpr, tpr, label='ROC curve')
plt.plot([0, 1], [0, 1], 'k--', label='Random guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.xlim([-0.02, 1])
plt.ylim([0, 1.02])
plt.legend(loc="lower right")
plt.show()
```



In [203]:
```
ras = roc_auc_score(Y_test, out_pred_prob)
ras
```

Out[203]: 0.7822222222222223

In [206]:
```
 precision, recall, thresholds = precision_recall_curve(Y_test, out_pred_prob)
# create plot
plt.plot(precision, recall, label='Precision-recall curve')
_ = plt.xlabel('Precision')
_ = plt.ylabel('Recall')
_ = plt.title('Precision-recall curve')
_ = plt.legend(loc="lower left")
```



In [207]:
```
aps = average_precision_score(Y_test, out_pred_prob)
aps
```

Out[207]:
```
0.7052309352299042
```

In [217]:
```

```

In [ ]:
```

```