```
!pip install seaborn

Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
     ---------------------------------------- 0.0/293.3 kB ? eta
-:--:--
     ---------------------------------------  286.7/293.3 kB 5.9 MB/s
eta 0:00:01
     ---------------------------------------- 293.3/293.3 kB 2.6 MB/s
eta 0:00:00
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\91805\
anaconda1\lib\site-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=0.25 in c:\users\91805\
anaconda1\lib\site-packages (from seaborn) (2.0.0)
Collecting matplotlib!=3.6.1,>=3.1
  Downloading matplotlib-3.7.1-cp39-cp39-win_amd64.whl (7.6 MB)
     ---------------------------------------- 0.0/7.6 MB ? eta -:--:--
     - -------------------------------------- 0.3/7.6 MB 8.6 MB/s eta
0:00:01
     ---- ----------------------------------- 0.9/7.6 MB 9.2 MB/s eta
0:00:01
     ------ --------------------------------- 1.2/7.6 MB 8.6 MB/s eta
0:00:01
     -------- ------------------------------- 1.7/7.6 MB 8.3 MB/s eta
0:00:01
     ----------- ---------------------------- 2.1/7.6 MB 8.0 MB/s eta
0:00:01
     ------------- -------------------------- 2.5/7.6 MB 8.1 MB/s eta
0:00:01
     -------------- ------------------------- 2.8/7.6 MB 7.9 MB/s eta
0:00:01
     ---------------- ----------------------- 3.2/7.6 MB 7.8 MB/s eta
0:00:01
     ------------------- -------------------- 3.7/7.6 MB 8.2 MB/s eta
0:00:01
     ----------------------------- ---------- 5.6/7.6 MB 8.6 MB/s eta
0:00:01
     ------------------------------- -------- 6.1/7.6 MB 8.5 MB/s eta
0:00:01
     ------------------------------- -------- 6.1/7.6 MB 8.3 MB/s eta
0:00:01
     ------------------------------- -------- 6.1/7.6 MB 8.3 MB/s eta
0:00:01
     ------------------------------------- -- 7.1/7.6 MB 8.2 MB/s eta
0:00:01
     --------------------------------------- - 7.3/7.6 MB 8.4 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
```

```
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------  7.6/7.6 MB 8.0 MB/s eta
0:00:01
     ---------------------------------------- 7.6/7.6 MB 5.7 MB/s eta
0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp39-cp39-win_amd64.whl (55 kB)
     ---------------------------------------- 0.0/55.4 kB ? eta
-:--:--
     ------------------------------------ --- 51.2/55.4 kB ? eta
-:--:--
     -------------------------------------- 55.4/55.4 kB 579.1 kB/s
eta 0:00:00
Requirement already satisfied: pillow>=6.2.0 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(9.4.0)
Collecting importlib-resources>=3.2.0
  Downloading importlib_resources-5.12.0-py3-none-any.whl (36 kB)
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.7-cp39-cp39-win_amd64.whl (160 kB)
     ---------------------------------------- 0.0/160.2 kB ? eta
-:--:--
     ------------------------------------- - 153.6/160.2 kB ? eta
-:--:--
     -------------------------------------- 160.2/160.2 kB 1.9 MB/s
eta 0:00:00
Collecting pyparsing>=2.3.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
     ---------------------------------------- 0.0/98.3 kB ? eta
-:--:--
     ------------------------------------ -- 92.2/98.3 kB ? eta
-:--:--
     ---------------------------------------- 98.3/98.3 kB 1.1 MB/s
eta 0:00:00
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(23.0)
```

```
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.39.4-cp39-cp39-win_amd64.whl (2.0 MB)
     ---------------------------------------- 0.0/2.0 MB ? eta -:--:--
     ---------- ----------------------------- 0.5/2.0 MB 31.4 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------- ------------------- 1.0/2.0 MB 16.7 MB/s eta
0:00:01
     -------------------------- ------------- 1.4/2.0 MB 2.3 MB/s eta
0:00:01
     ---------------------------------- ----- 1.7/2.0 MB 2.6 MB/s eta
0:00:01
     ---------------------------------------  2.0/2.0 MB 2.8 MB/s eta
0:00:01
     ---------------------------------------  2.0/2.0 MB 2.8 MB/s eta
0:00:01
     ---------------------------------------- 2.0/2.0 MB 2.5 MB/s eta
0:00:00
Requirement already satisfied: tzdata>=2022.1 in c:\users\91805\
anaconda1\lib\site-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\91805\
anaconda1\lib\site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: zipp>=3.1.0 in c:\users\91805\
anaconda1\lib\site-packages (from importlib-resources>=3.2.0-
>matplotlib!=3.6.1,>=3.1->seaborn) (3.11.0)
Requirement already satisfied: six>=1.5 in c:\users\91805\anaconda1\
lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1-
>seaborn) (1.16.0)
Installing collected packages: pyparsing, kiwisolver, importlib-
```

```
resources, fonttools, cycler, contourpy, matplotlib, seaborn
Successfully installed contourpy-1.0.7 cycler-0.11.0 fonttools-4.39.4
importlib-resources-5.12.0 kiwisolver-1.4.4 matplotlib-3.7.1
pyparsing-3.0.9 seaborn-0.12.2
```

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\91805\anaconda1\
lib\site-packages (3.7.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (4.39.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (1.0.7)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\
91805\anaconda1\lib\site-packages (from matplotlib) (5.12.0)
Requirement already satisfied: cycler>=0.10 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: packaging>=20.0 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: numpy>=1.20 in c:\users\91805\
anaconda1\lib\site-packages (from matplotlib) (1.23.5)
Requirement already satisfied: zipp>=3.1.0 in c:\users\91805\
anaconda1\lib\site-packages (from importlib-resources>=3.2.0-
>matplotlib) (3.11.0)
Requirement already satisfied: six>=1.5 in c:\users\91805\anaconda1\
lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
!pip install imbalanced-learn
```

```
Collecting imbalanced-learn
  Using cached imbalanced_learn-0.10.1-py3-none-any.whl (226 kB)
Requirement already satisfied: joblib>=1.1.1 in c:\users\91805\
anaconda1\lib\site-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\91805\
anaconda1\lib\site-packages (from imbalanced-learn) (1.23.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\91805\
anaconda1\lib\site-packages (from imbalanced-learn) (2.2.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\91805\
anaconda1\lib\site-packages (from imbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\91805\
anaconda1\lib\site-packages (from imbalanced-learn) (1.2.2)
Installing collected packages: imbalanced-learn
Successfully installed imbalanced-learn-0.10.1
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, recall_score, confusion_matrix,
roc_auc_score, roc_curve, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

import warnings
warnings.filterwarnings('ignore')
```

```python
pwd
```

```
'C:\\Users\\91805'
```

```python
df = pd.read_csv('health care diabetes.csv')
```

```python
df
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI

0              6      148             72             35        0  33.6
\
1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

763           10      101             76             48      180  32.9

764            2      122             70             27        0  36.8

765            5      121             72             23      112  26.2

766            1      126             60              0        0  30.1

767            1       93             70             31        0  30.4
```

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

[768 rows x 9 columns]
```

```
df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI

0            6      148             72             35        0
33.6  \
1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

1.Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value: • Glucose

• BloodPressure

• SkinThickness

• Insulin

• BMI

2.Visually explore these variables using histograms. Treat the missing values accordingly.

3.There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

```
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
```

```
df.isna().sum()
```

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

```
df.shape
```

```
(768, 9)
```

```
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness   
Insulin  
count   768.000000  768.000000     768.000000     768.000000  
768.000000  \
mean      3.845052  120.894531      69.105469      20.536458   
79.799479   
std       3.369578   31.972618      19.355807      15.952218  
115.244002   
min       0.000000    0.000000       0.000000       0.000000
```

```
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

• This Datasets have 9 variables and 768 Observations

• The dataset helps to predict the diabetes of various age group of women using the variables of pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin and BMI.

• The Average Age of Patients are 33.24 with minimum being 21 and maximum 81

```
df.isnull().any()
```

```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

```
df.isna().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
```

```
Outcome                     0
dtype: int64
```

```
df.columns
```

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

```
df.isna().sum(axis=1).max()
```

```
0
```

```
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness
Insulin
count   768.000000  768.000000     768.000000     768.000000
768.000000  \
mean      3.845052  120.894531      69.105469      20.536458
79.799479
std       3.369578   31.972618      19.355807      15.952218
115.244002
min       0.000000    0.000000       0.000000       0.000000
0.000000
25%       1.000000   99.000000      62.000000       0.000000
0.000000
50%       3.000000  117.000000      72.000000      23.000000
30.500000
75%       6.000000  140.250000      80.000000      32.000000
127.250000
max      17.000000  199.000000     122.000000      99.000000
846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

```
print((df[['Glucose']]==0).sum())
```

```
Glucose    5
dtype: int64
```

```
print((df[['BloodPressure']]==0).sum())
```

```
BloodPressure    35
dtype: int64
```

```python
print((df[['SkinThickness']]==0).sum())
```

```
SkinThickness    227
dtype: int64
```

```python
print((df[['Insulin']]==0).sum())
```

```
Insulin    374
dtype: int64
```

```python
print((df[['BMI']]==0).sum())
```

```
BMI    11
dtype: int64
```

```python
print ((df[['Pregnancies', 'Glucose', 'BloodPressure', 
'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']] == 0).sum())
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
dtype: int64
```

```python
print((df[['Glucose']]==0).count())
```

```
Glucose    768
dtype: int64
```

```python
print ((df[['Pregnancies', 'Glucose', 'BloodPressure', 
'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']] == 0).count())
```

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
dtype: int64
```

```python
df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI

0            6      148             72             35        0
33.6  \
1            1       85             66             29        0  26.6

2            8      183             64              0        0  23.3

3            1       89             66             23       94  28.1

4            0      137             40             35      168  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
df [['Pregnancies','Glucose', 'BloodPressure', 'SkinThickness', 
'Insulin',
       'BMI', ]] = df [['Pregnancies','Glucose', 'BloodPressure', 
'SkinThickness', 'Insulin',
       'BMI', ]].replace(0,np.NaN)
```

```
df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI

0          6.0    148.0           72.0           35.0      NaN
33.6  \
1          1.0     85.0           66.0           29.0      NaN  26.6

2          8.0    183.0           64.0            NaN      NaN  23.3

3          1.0     89.0           66.0           23.0     94.0  28.1

4          NaN    137.0           40.0           35.0    168.0  43.1


   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
df['Pregnancies'].fillna(df['Pregnancies'].mean(), inplace = True)
```

```
print(df['Pregnancies'].isnull().sum())
```

```
0
```

```
df.fillna(df.mean(), inplace=True)
```

```
df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin
BMI
0     6.000000    148.0           72.0       35.00000  155.548223
33.6  \
1     1.000000     85.0           66.0       29.00000  155.548223
26.6
2     8.000000    183.0           64.0       29.15342  155.548223
23.3
3     1.000000     89.0           66.0       23.00000   94.000000
28.1
4     4.494673    137.0           40.0       35.00000  168.000000
43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

```
print (df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']].isnull().sum())
```

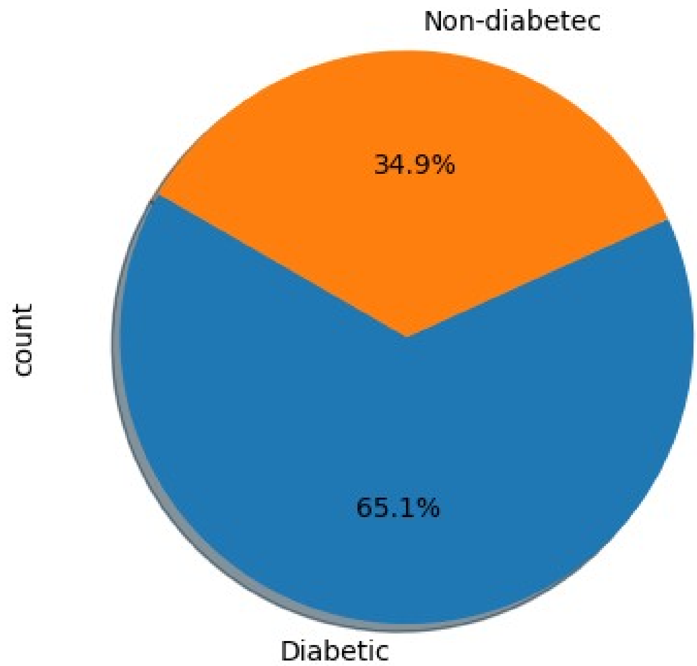
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
dtype: int64
```

```
df.groupby('Outcome').size()
```

```
Outcome
0    500
1    268
dtype: int64
```

```
labels = 'Diabetic', 'Non-diabetec'
df.Outcome.value_counts().plot.pie(labels=labels, autopct='%1.1f%
%',shadow=True, startangle=150)
```

```
<Axes: ylabel='count'>
```



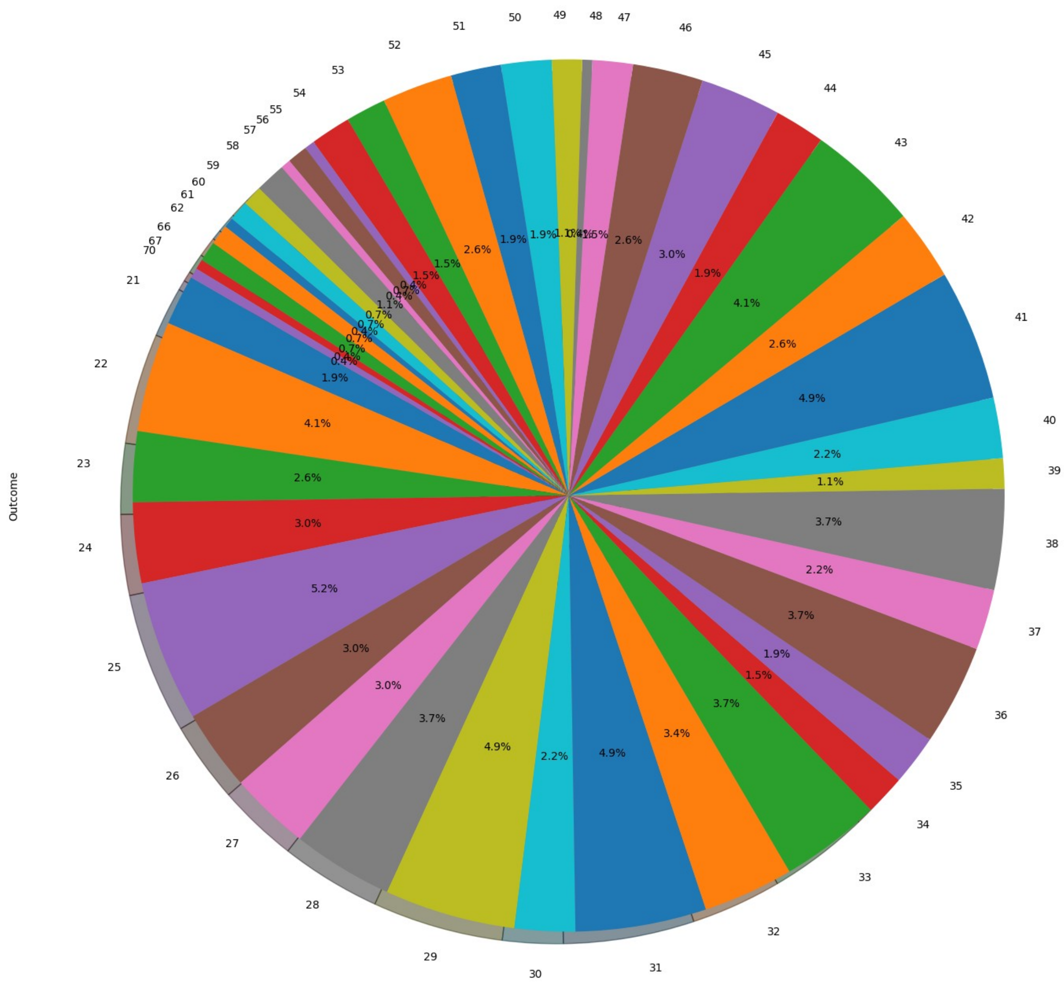

```
diabetes_agewise = df[df['Outcome']==1]
diabetes_agewise.groupby('Age')['Outcome'].count()
```

```
Age
21     5
22    11
23     7
24     8
25    14
26     8
27     8
28    10
29    13
30     6
31    13
32     9
33    10
34     4
35     5
36    10
37     6
38    10
39     3
40     6
41    13
42     7
```

```
43    11
44     5
45     8
46     7
47     4
48     1
49     3
50     5
51     5
52     7
53     4
54     4
55     1
56     2
57     1
58     3
59     2
60     2
61     1
62     2
66     2
67     1
70     1
Name: Outcome, dtype: int64
```

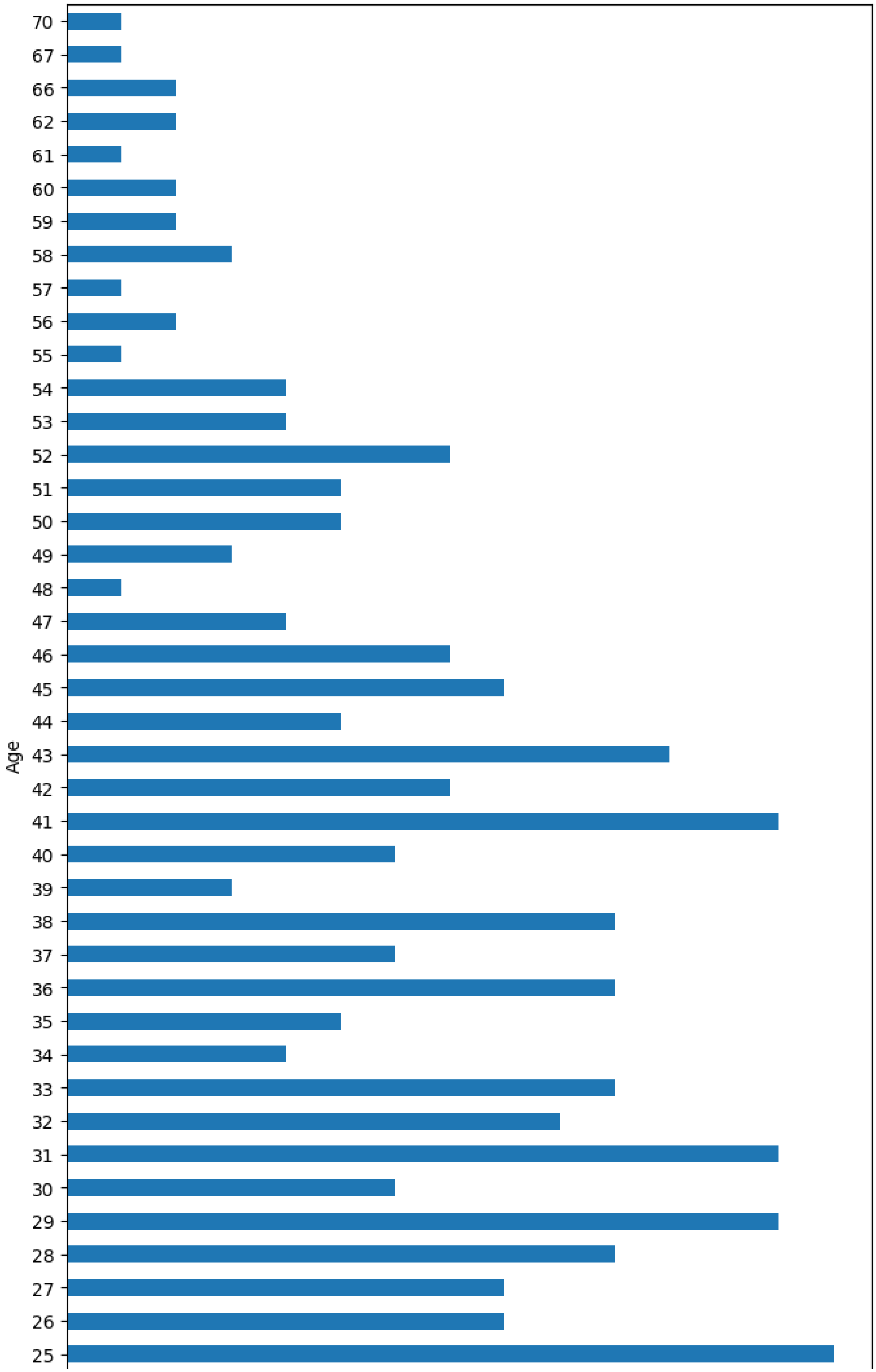
```
diabetes_agewise.groupby('Age')
['Outcome'].count().plot.pie(autopct='%1.1f%%',shadow=True,
startangle=150, figsize=(35,18))
```

```
<Axes: ylabel='Outcome'>
```



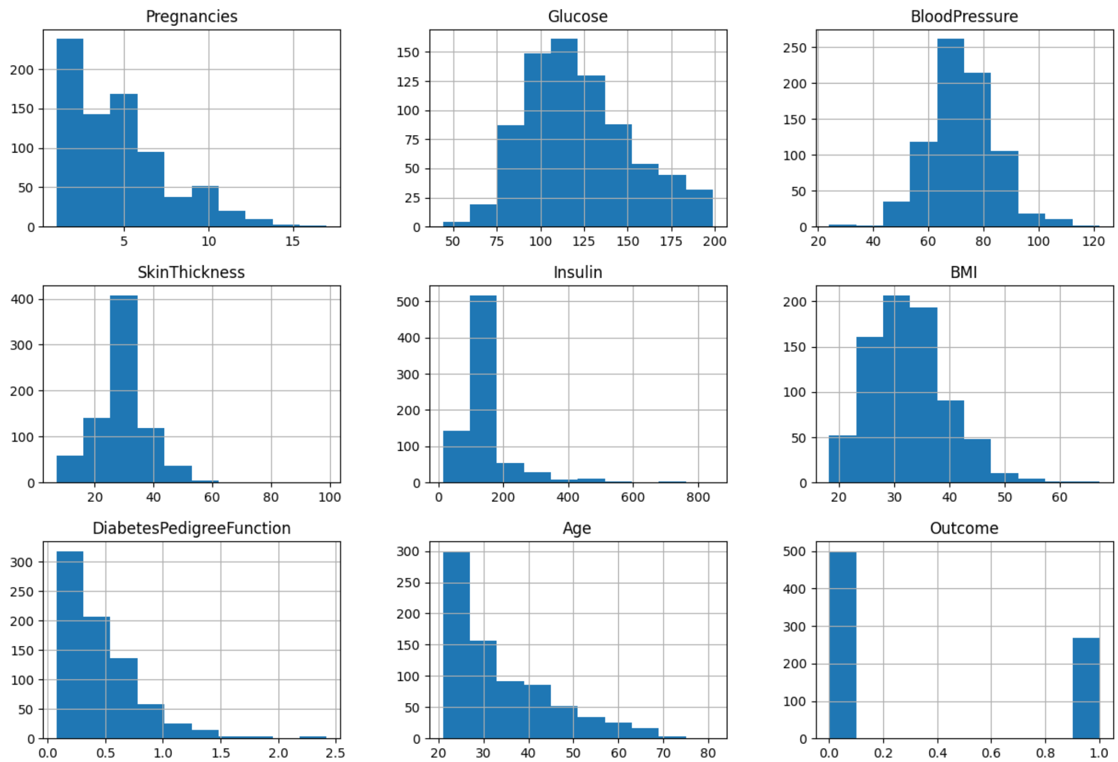

```
diabetes_agewise.groupby('Age')['Outcome'].count().plot(kind= 'barh',
figsize=(8,15))
```

```
<Axes: ylabel='Age'>
```


Age
70
67
66
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25

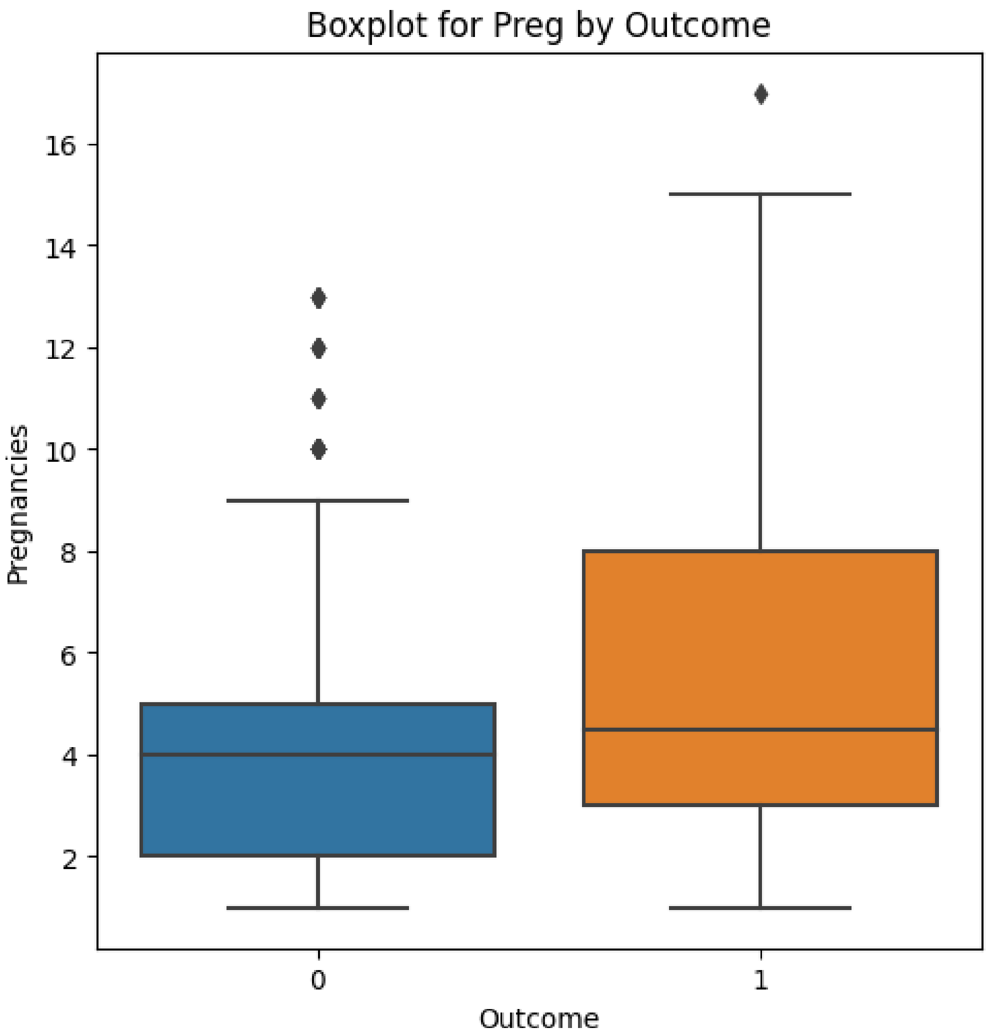
```
df.hist(figsize=(15,10))
```

```
array([[<Axes: title={'center': 'Pregnancies'}>,
        <Axes: title={'center': 'Glucose'}>,
        <Axes: title={'center': 'BloodPressure'}>],
       [<Axes: title={'center': 'SkinThickness'}>,
        <Axes: title={'center': 'Insulin'}>,
        <Axes: title={'center': 'BMI'}>],
       [<Axes: title={'center': 'DiabetesPedigreeFunction'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Outcome'}>]], dtype=object)
```



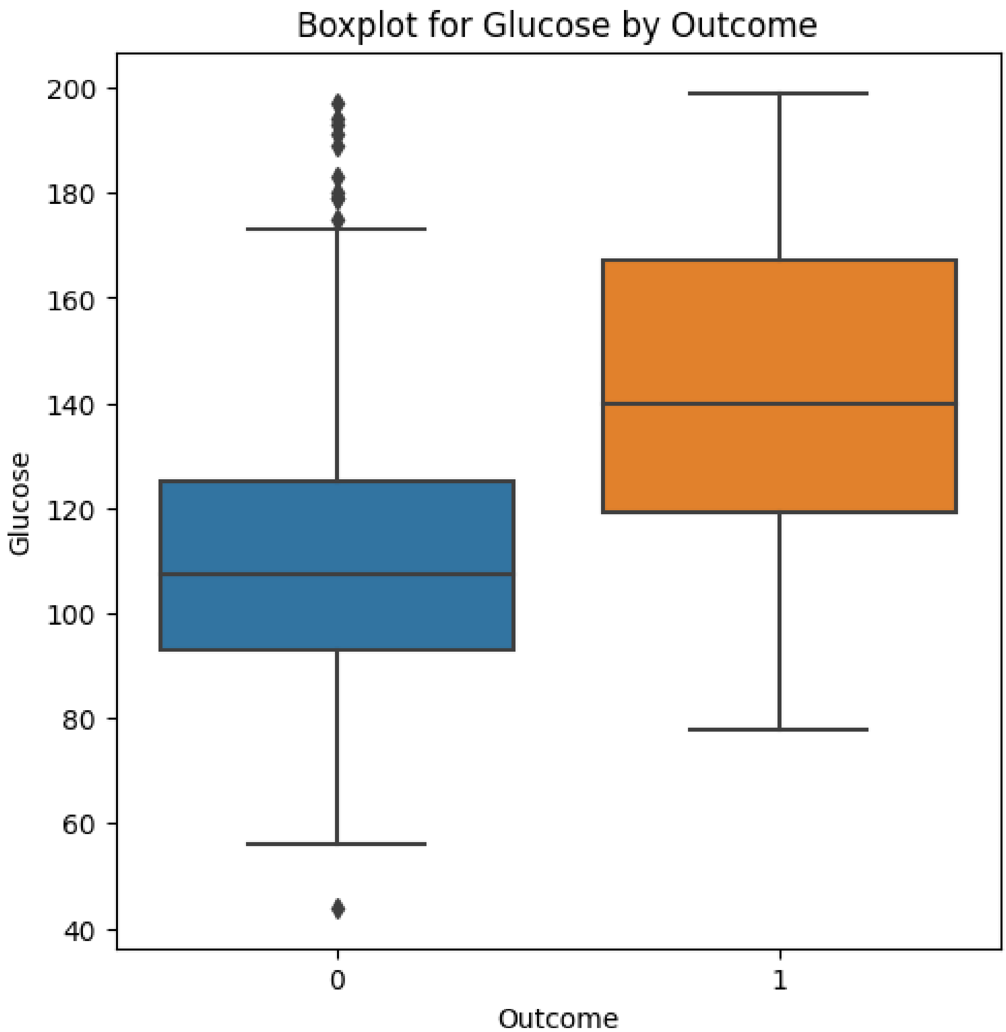

```
# Plots for count of outcome by values
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Pregnancies)
plt.title("Boxplot for Preg by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Preg by Outcome')
```



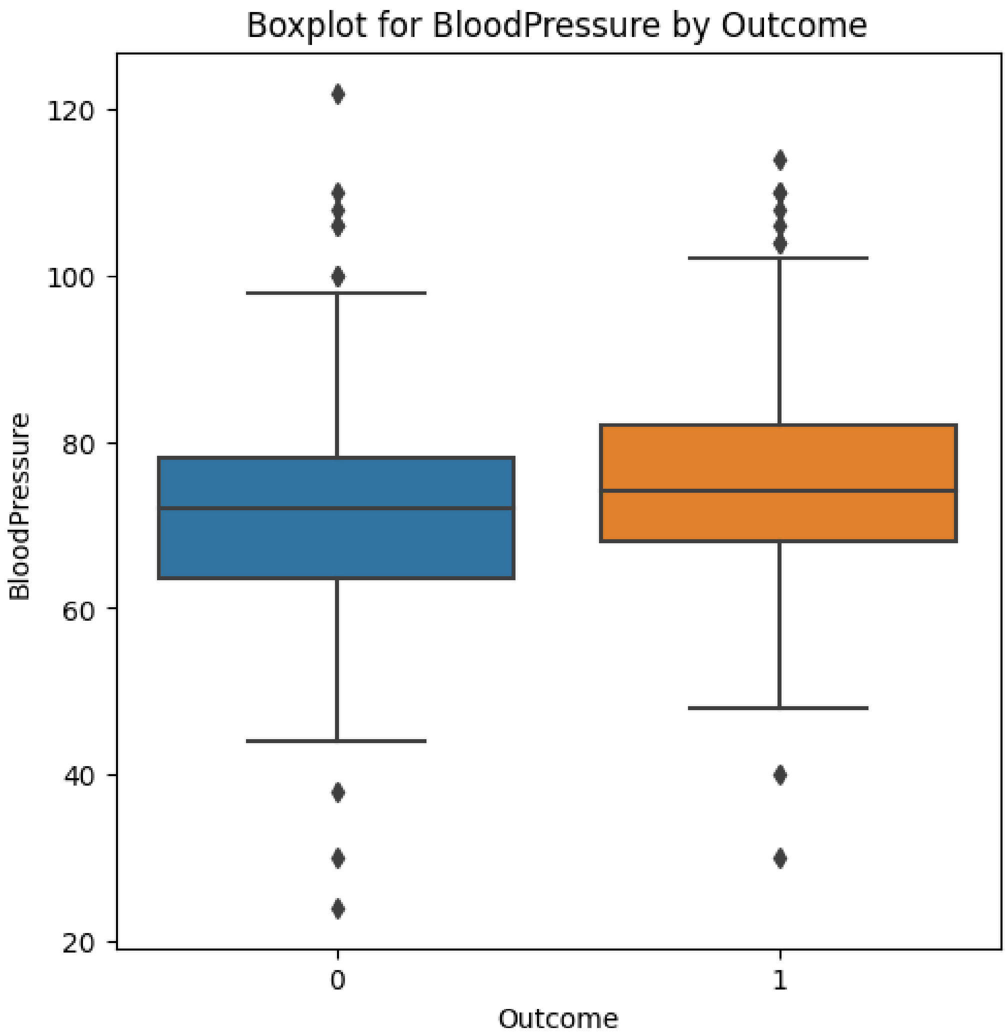

```
# Plot for glucose
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Glucose)
plt.title("Boxplot for Glucose by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Glucose by Outcome')
```



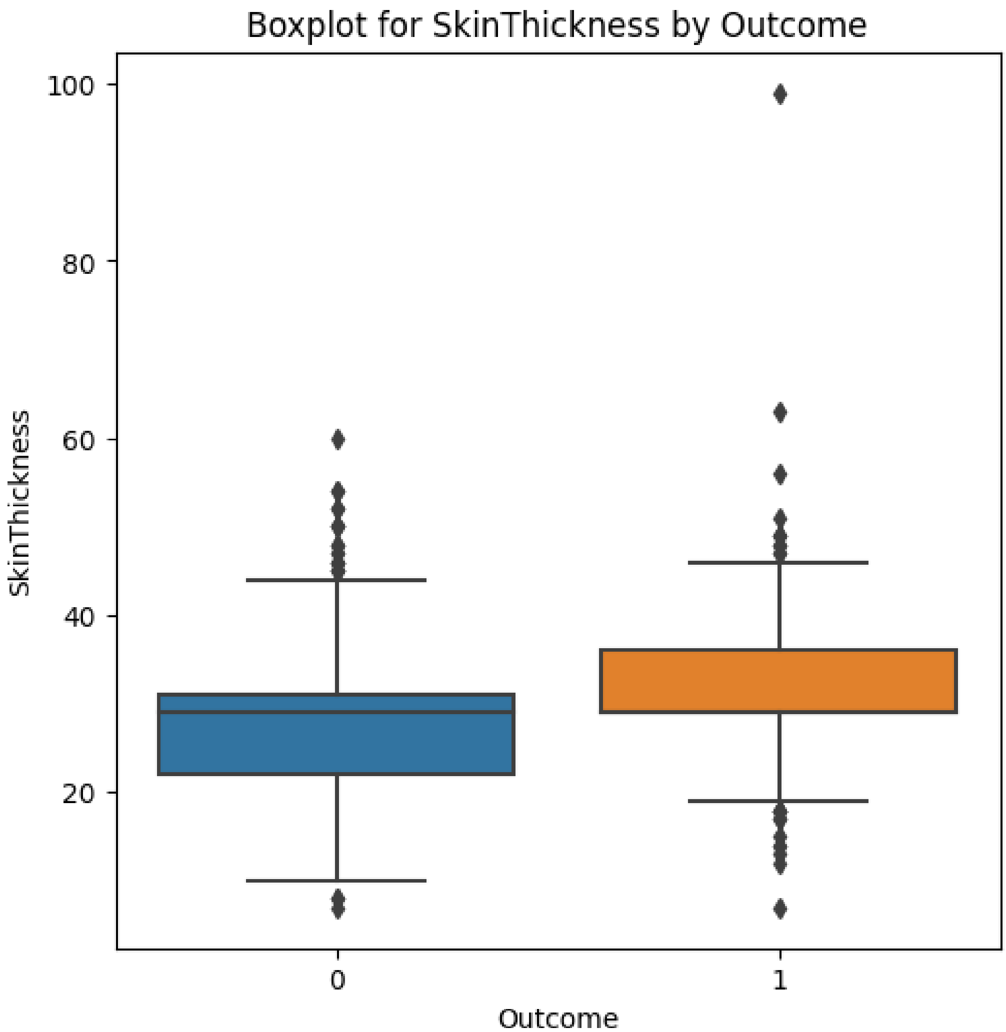

```
# Plot for BloodPressure
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.BloodPressure)
plt.title("Boxplot for BloodPressure by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for BloodPressure by Outcome')
```



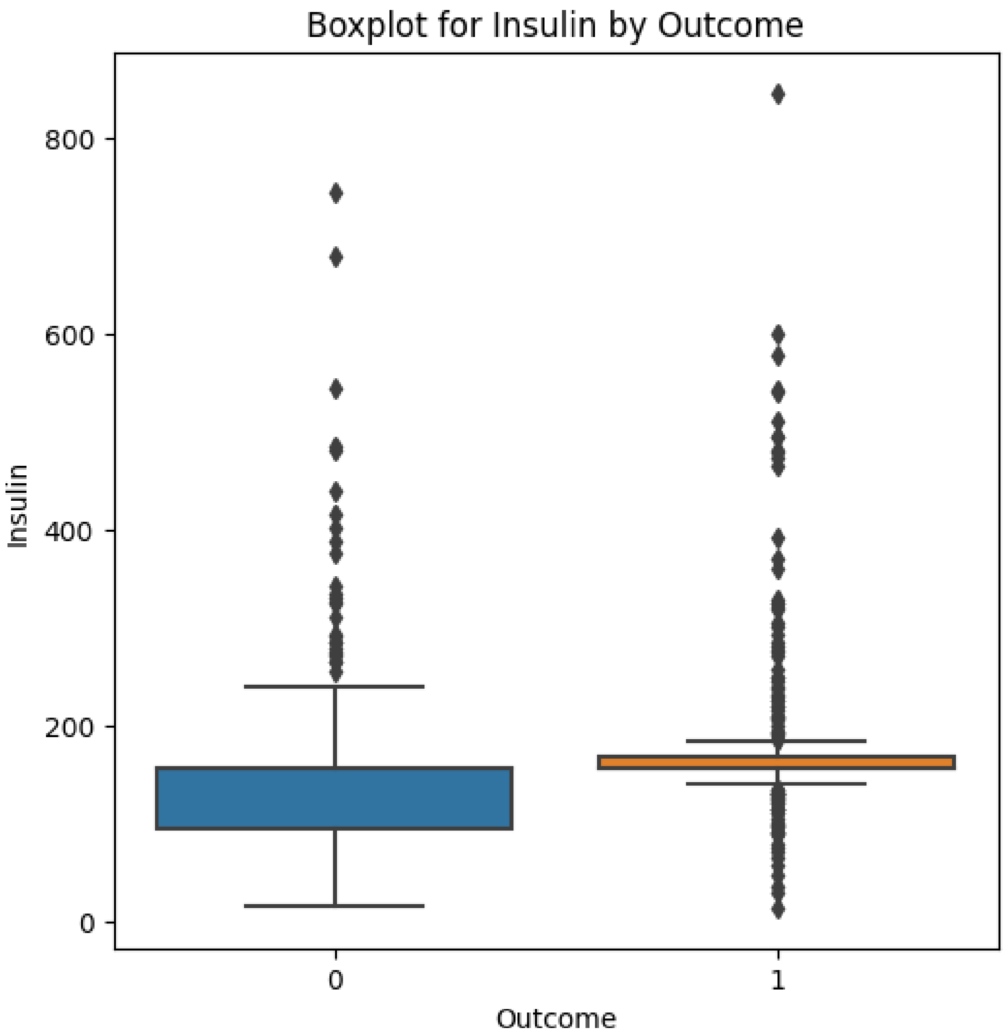

```
# Plot for SkinThickness
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.SkinThickness)
plt.title("Boxplot for SkinThickness by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for SkinThickness by Outcome')
```



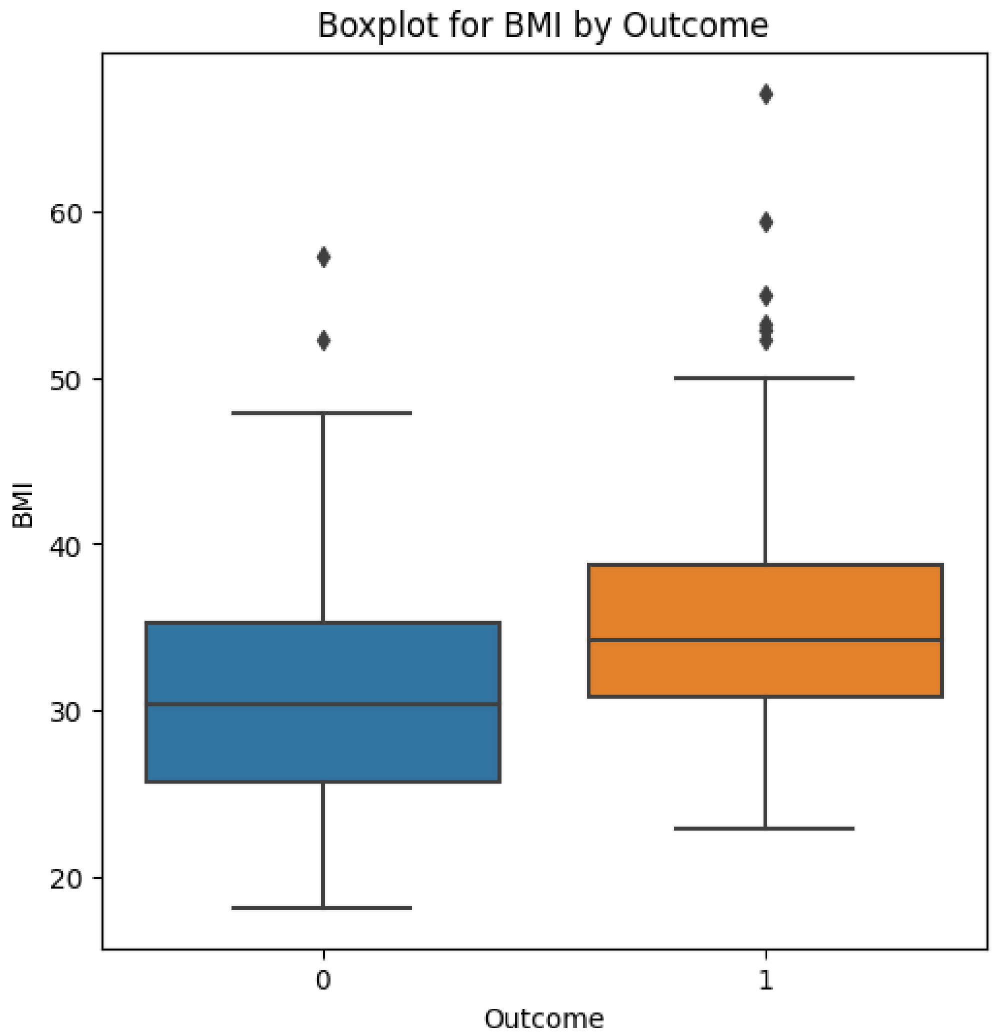

```
# plot for Insulin
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Insulin)
plt.title("Boxplot for Insulin by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Insulin by Outcome')
```



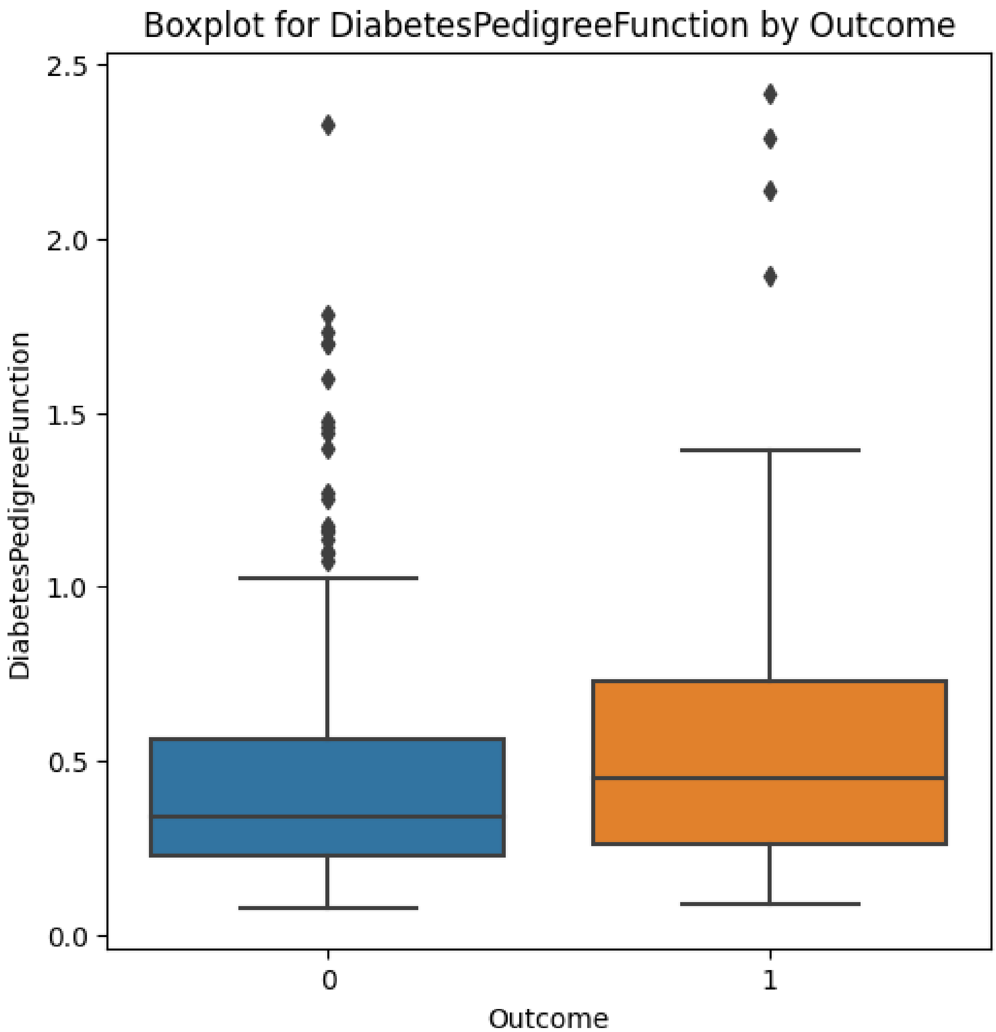

```
# Plot for BMI
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.BMI)
plt.title("Boxplot for BMI by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for BMI by Outcome')
```



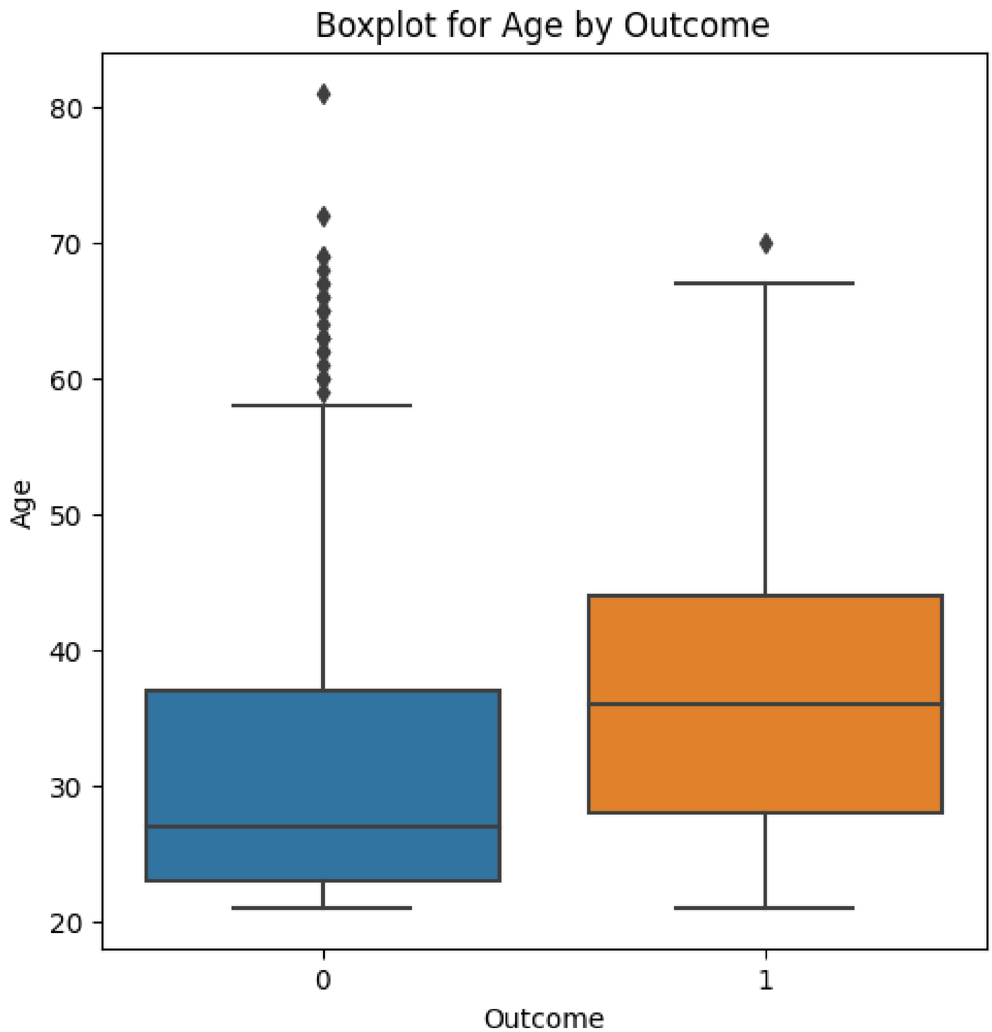

```
# Plot for Diabetes Pedigree Function
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.DiabetesPedigreeFunction)
plt.title("Boxplot for DiabetesPedigreeFunction by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for DiabetesPedigreeFunction by Outcome')
```



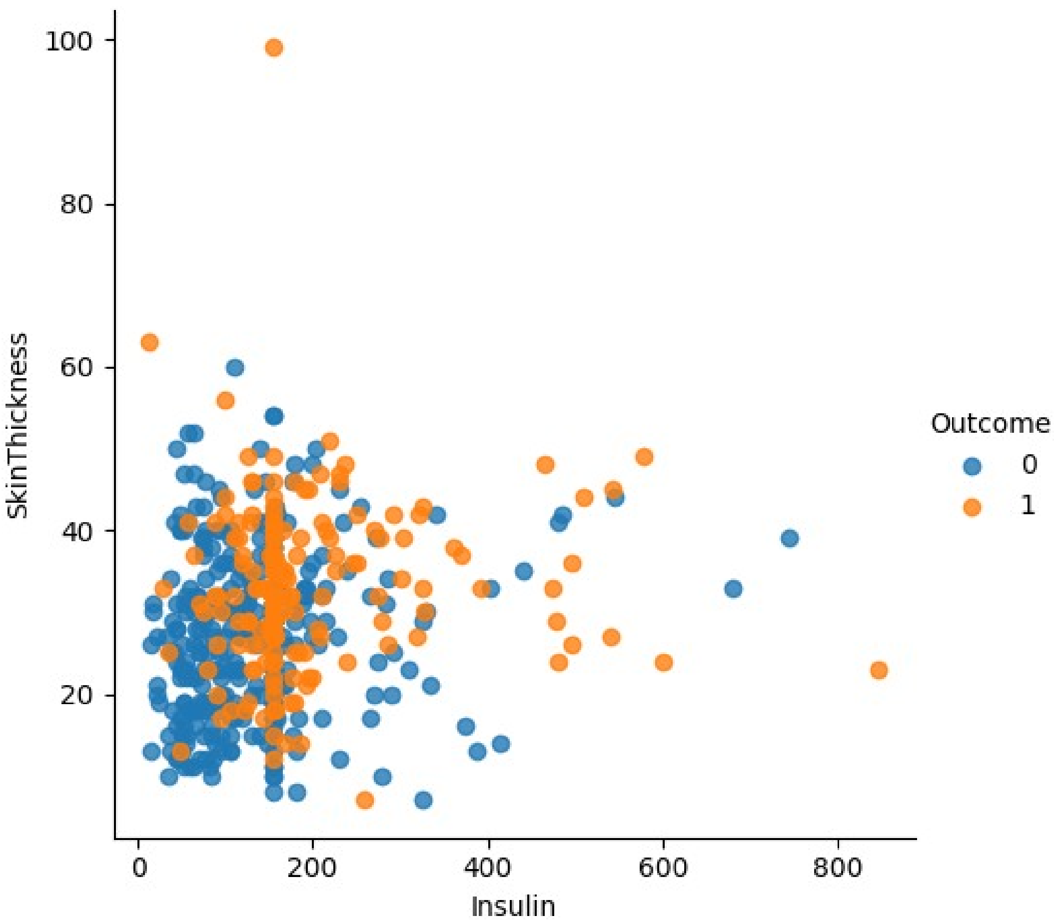

```python
# Plot for Age
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Age)
plt.title("Boxplot for Age by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Age by Outcome')
```



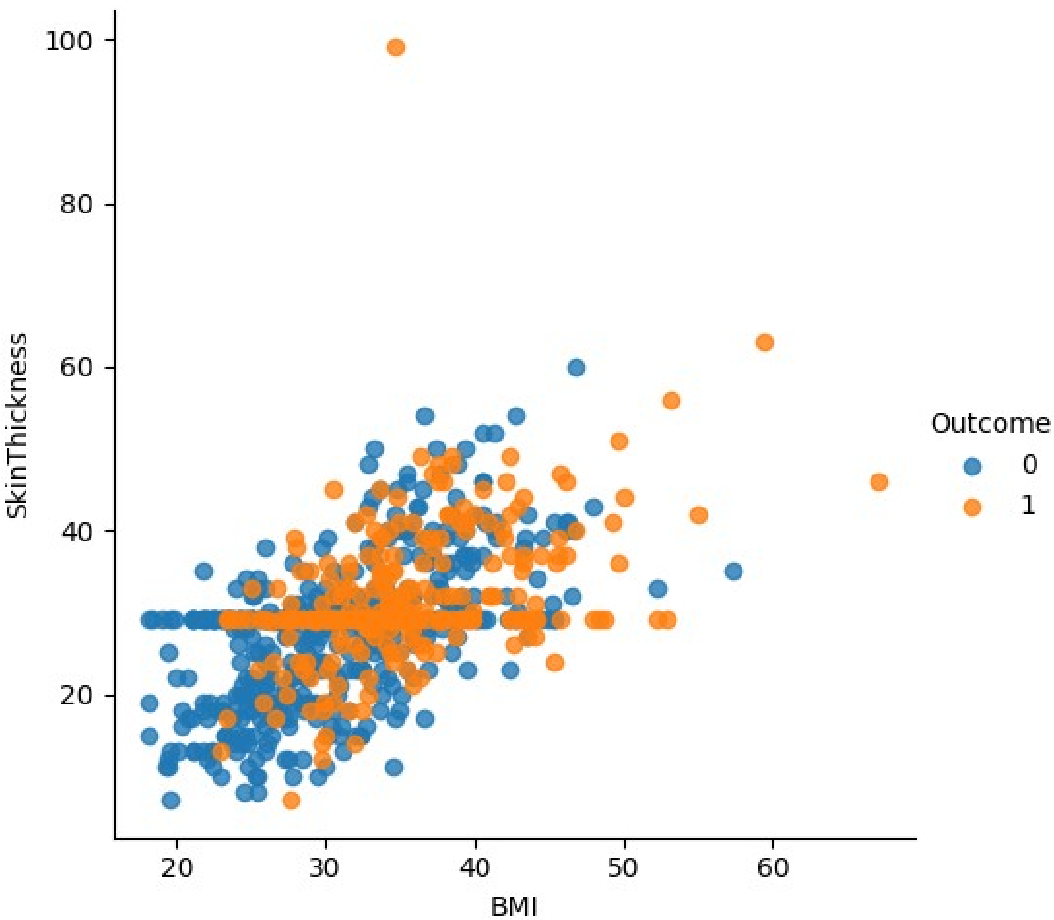

```
# Plot with outcome and variables
sns.lmplot(x='Insulin',y='SkinThickness',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae273e0d90>
```



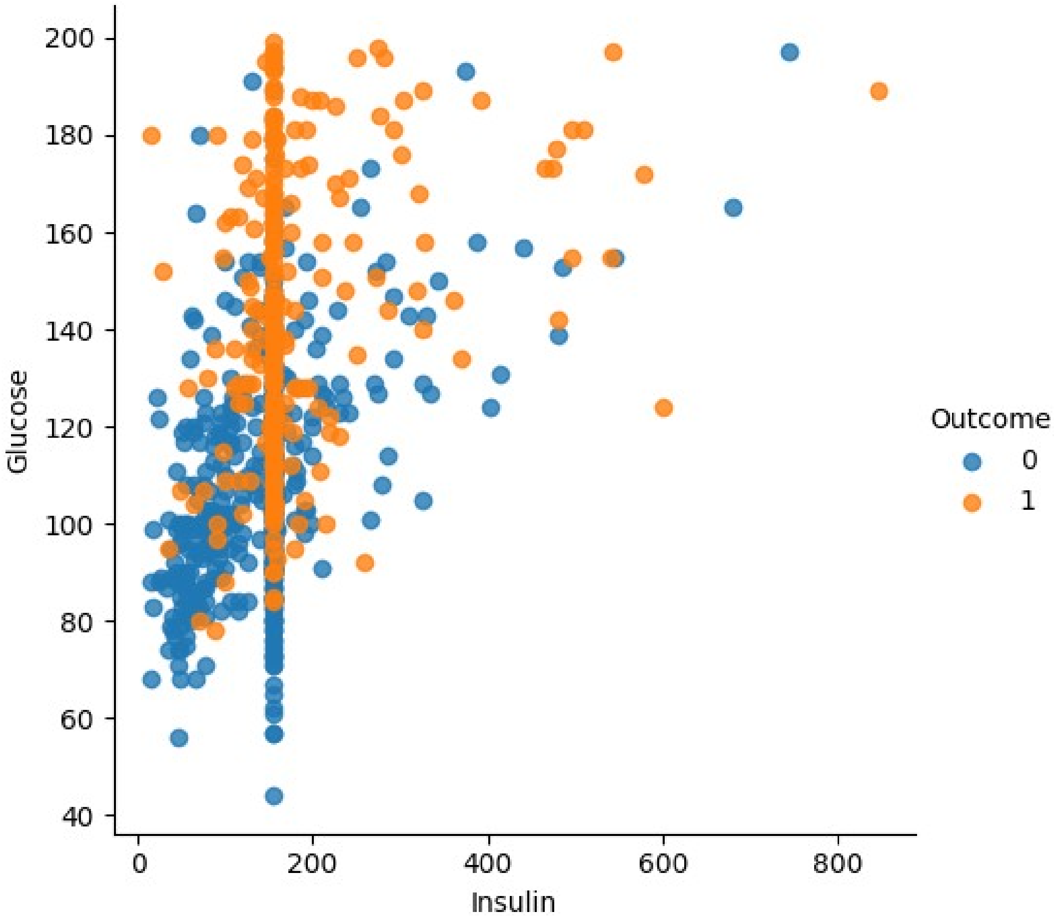

```
sns.lmplot(x='BMI',y='SkinThickness',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae24fefe50>
```



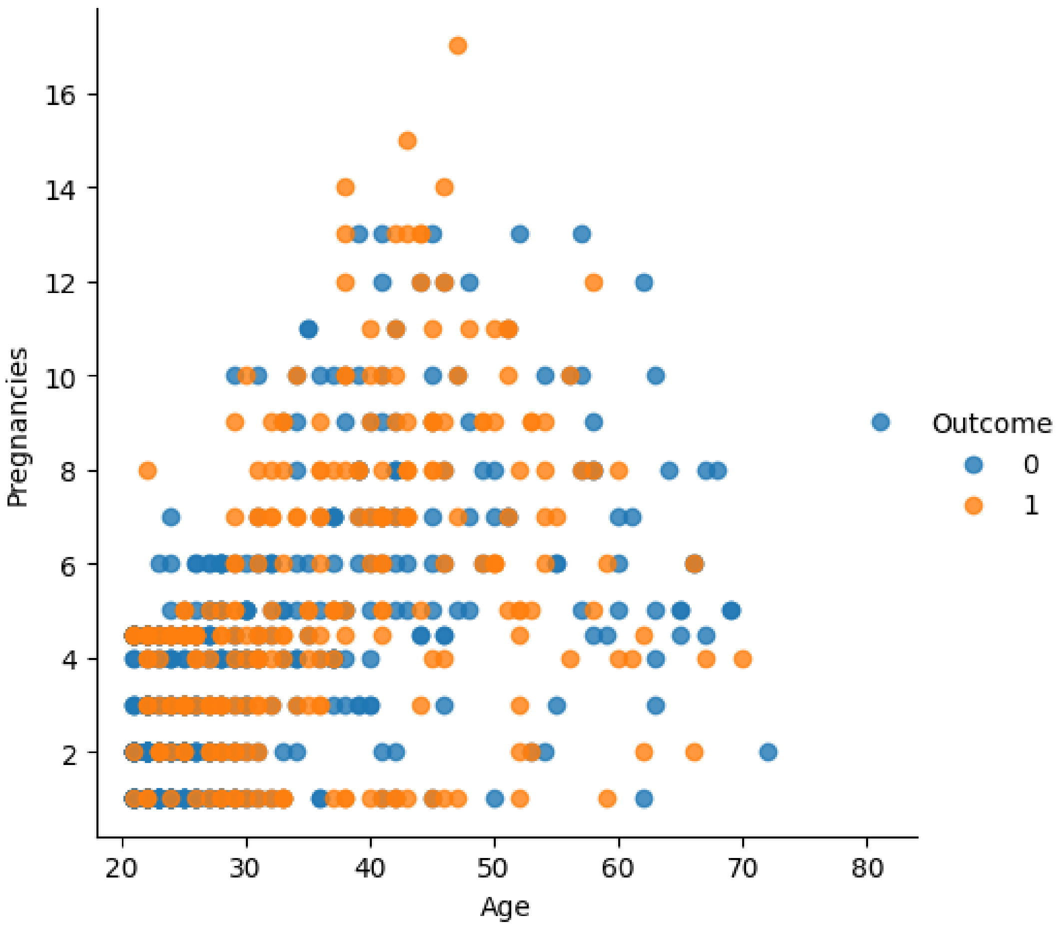

```
sns.lmplot(x='Insulin',y='Glucose',data=df,fit_reg=False,hue='Outcome'
)
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae27bcaac0>
```



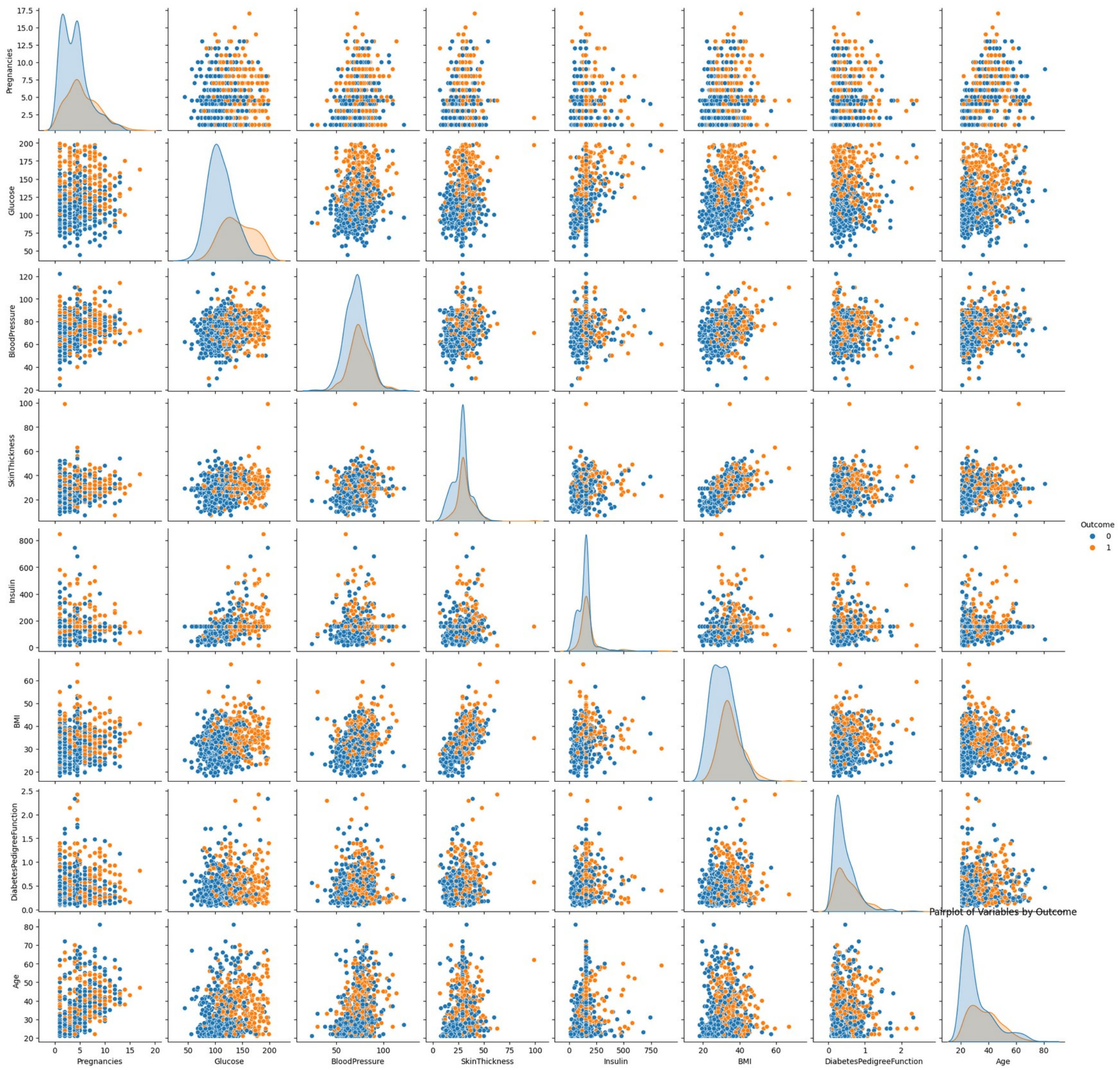

```
sns.lmplot(x='Age',y='Pregnancies',data=df,fit_reg=False,hue='Outcome'
)
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae28027d00>
```




```
sns.pairplot(df, vars=["Pregnancies",
"Glucose","BloodPressure","SkinThickness","Insulin",
"BMI","DiabetesPedigreeFunction", "Age"],hue="Outcome")
plt.title("Pairplot of Variables by Outcome")
```

```
Text(0.5, 1.0, 'Pairplot of Variables by Outcome')
```

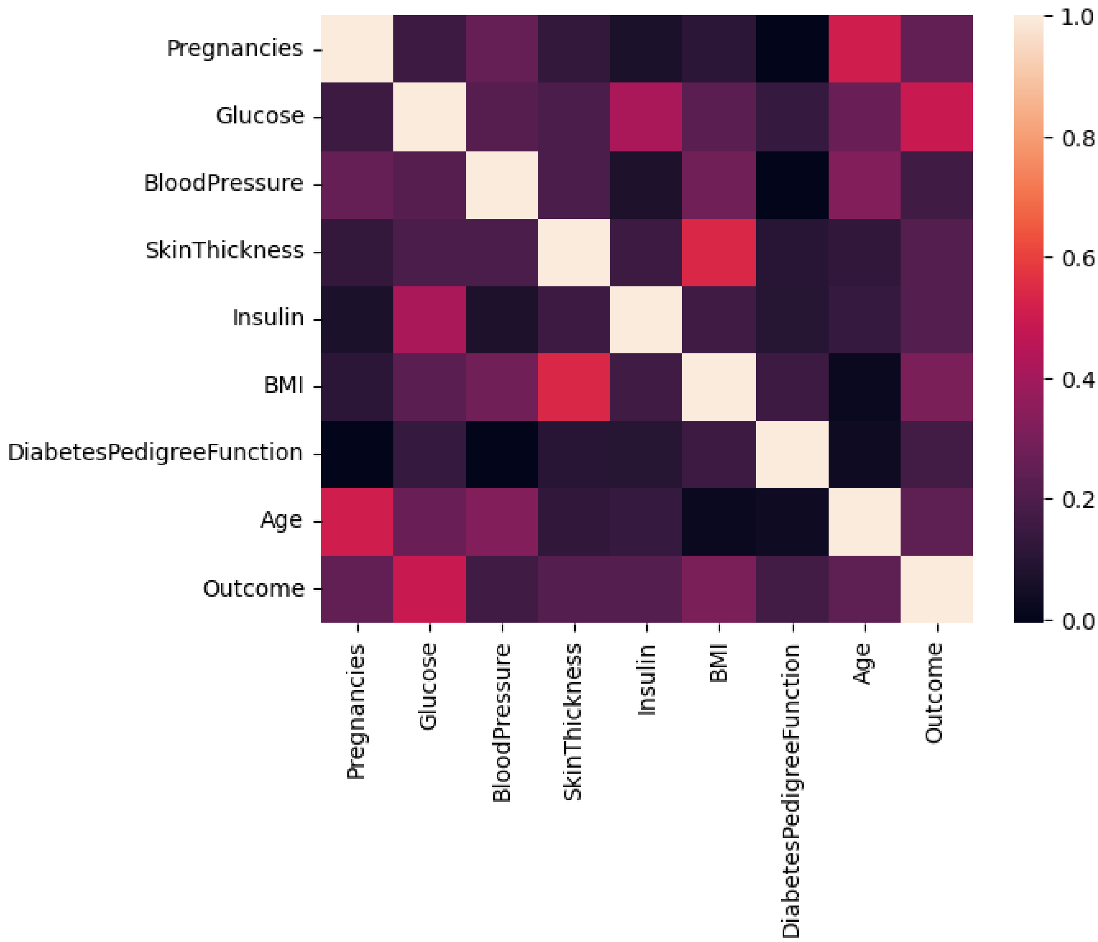



```
cor = df.corr()
cor

                           Pregnancies   Glucose  BloodPressure
SkinThickness
Pregnancies                   1.000000  0.154290       0.259117
0.131819  \
Glucose                       0.154290  1.000000       0.218367
0.192991
BloodPressure                 0.259117  0.218367       1.000000
0.192816
SkinThickness                 0.131819  0.192991       0.192816
1.000000
Insulin                       0.068077  0.420157       0.072517
0.158139
BMI                           0.110590  0.230941       0.281268
0.542398
DiabetesPedigreeFunction     -0.005658  0.137060      -0.002763
```

```
0.100966
Age                          0.511662  0.266534       0.324595
0.127872
Outcome                      0.248263  0.492928       0.166074
0.215299

                          Insulin       BMI  DiabetesPedigreeFunction

Pregnancies              0.068077  0.110590                 -0.005658
\
Glucose                  0.420157  0.230941                  0.137060

BloodPressure            0.072517  0.281268                 -0.002763

SkinThickness            0.158139  0.542398                  0.100966

Insulin                  1.000000  0.166586                  0.098634

BMI                      0.166586  1.000000                  0.153400

DiabetesPedigreeFunction 0.098634  0.153400                  1.000000

Age                      0.136734  0.025519                  0.033561

Outcome                  0.214411  0.311924                  0.173844


                              Age   Outcome
Pregnancies              0.511662  0.248263
Glucose                  0.266534  0.492928
BloodPressure            0.324595  0.166074
SkinThickness            0.127872  0.215299
Insulin                  0.136734  0.214411
BMI                      0.025519  0.311924
DiabetesPedigreeFunction 0.033561  0.173844
Age                      1.000000  0.238356
Outcome                  0.238356  1.000000
```

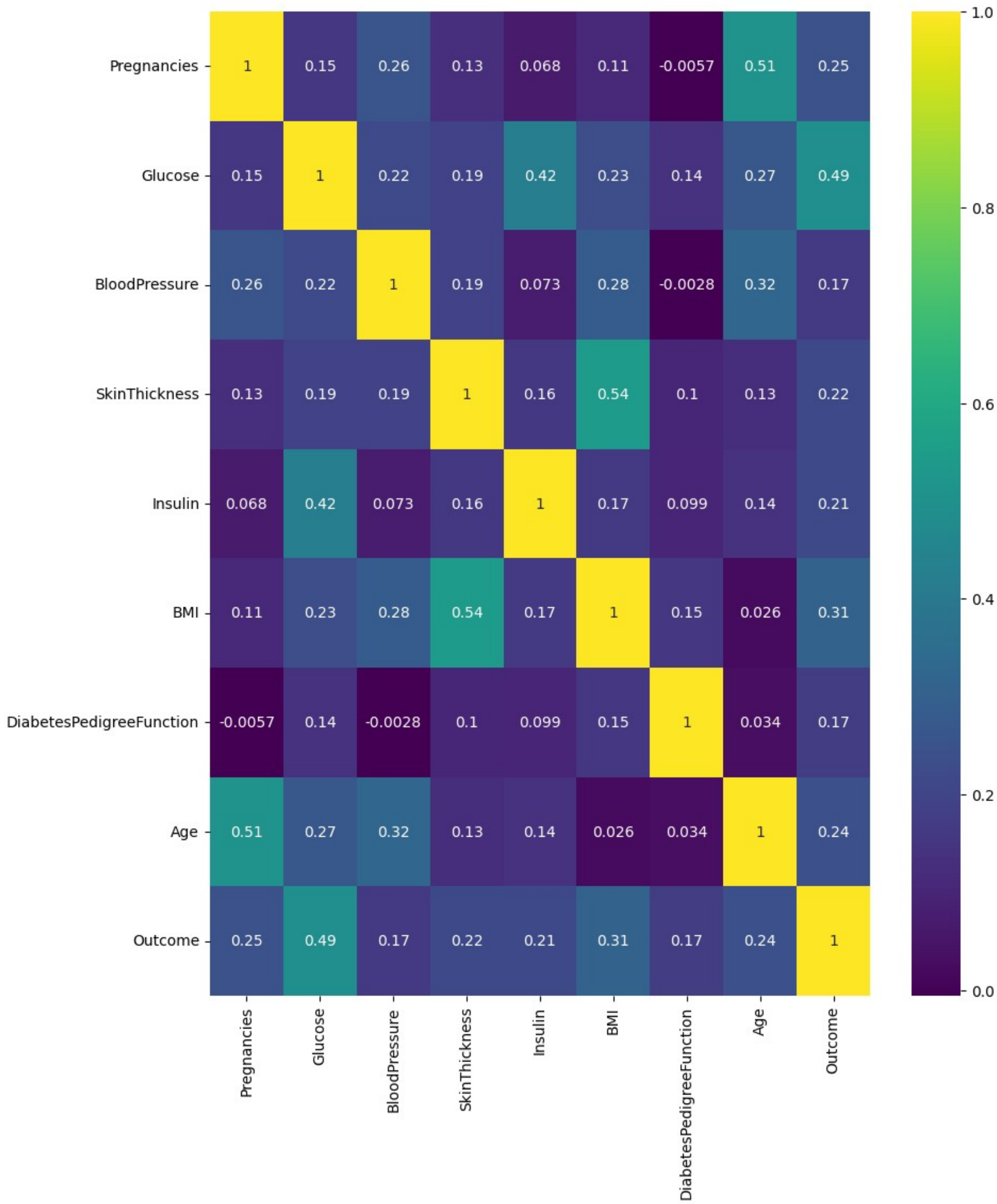
```
sns.heatmap(cor)
```

```
<Axes: >
```




```
plt.subplots(figsize=(10,12))
sns.heatmap(cor,annot=True,cmap='viridis')
```

```
<Axes: >
```




## Data Modeling:

Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.¶

Since it's a classification problem, we'll be building models using following classification algorithms for our training data and then compare performance of each model on test data to accurately predict target variable (Outcome):

1.Logistic Regression

2.Support Vector Machine (SVM)

3.K-Nearest Neighbour (KNN)

4.Decision Tree

5.RandomForest Classifier

6.Ensemble Learning -> Boosting -> Gradient Boosting (XGBClassifier)

```
features = df.iloc[:,[0,1,2,3,4,5,6,7]].values
label = df.iloc[:,8].values
```

```
#Train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(features,
                                                label,
                                                test_size=0.2,
                                                random_state =10)
```

```
#Create model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

model.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7850162866449512
0.7337662337662337
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

```
array([[448,  52],
       [121, 147]], dtype=int64)
```

```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

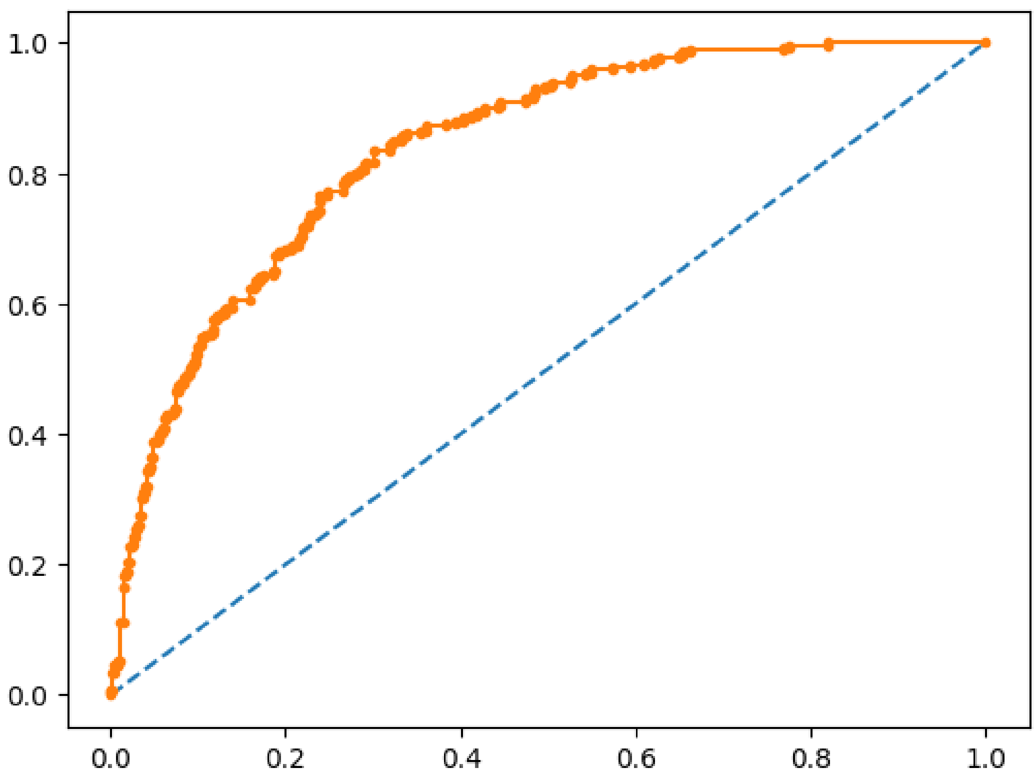
```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       500
           1       0.74      0.55      0.63       268

    accuracy                           0.77       768
   macro avg       0.76      0.72      0.73       768
```

```
weighted avg       0.77      0.77      0.77       768
```

```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.839
```

```
[<matplotlib.lines.Line2D at 0x1ae2dcaaa00>]
```




```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
```

```python
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=5)
```

```python
model3.score(X_train,y_train)
```

```
0.8208469055374593
```

```python
model3.score(X_test,y_test)
```

```
0.7532467532467533
```

```python
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

```
RandomForestClassifier(n_estimators=11)
```

```python
model4.score(X_train,y_train)
```

```
0.993485342019544
```

```python
model4.score(X_test,y_test)
```

```
0.7727272727272727
```

```python
#Support Vector Classifier

from sklearn.svm import SVC
model5 = SVC(kernel='rbf',
           gamma='auto')
model5.fit(X_train,y_train)
```

```
SVC(gamma='auto')
```

```python
model5.score(X_test,y_test)
```

```
0.6168831168831169
```

```python
model5.score(X_test,y_test)
```

```
0.6168831168831169
```

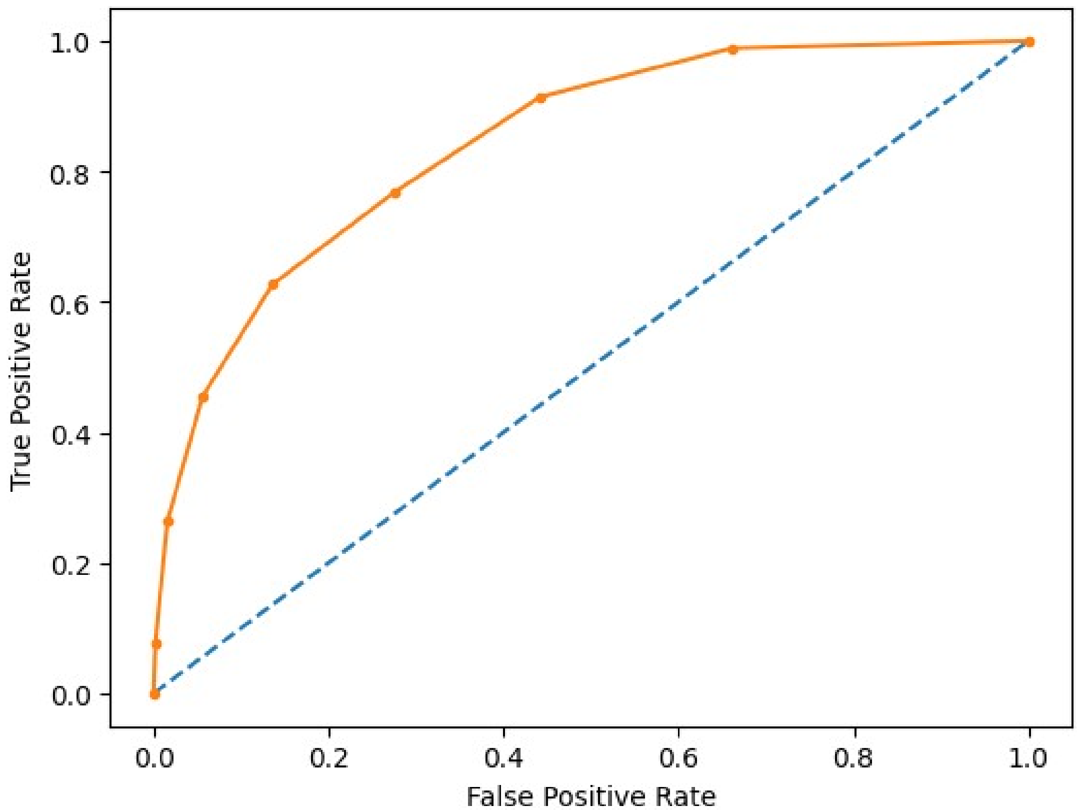
```python
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
model2.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```python
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds -
{}".format(tpr,fpr,thresholds))
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

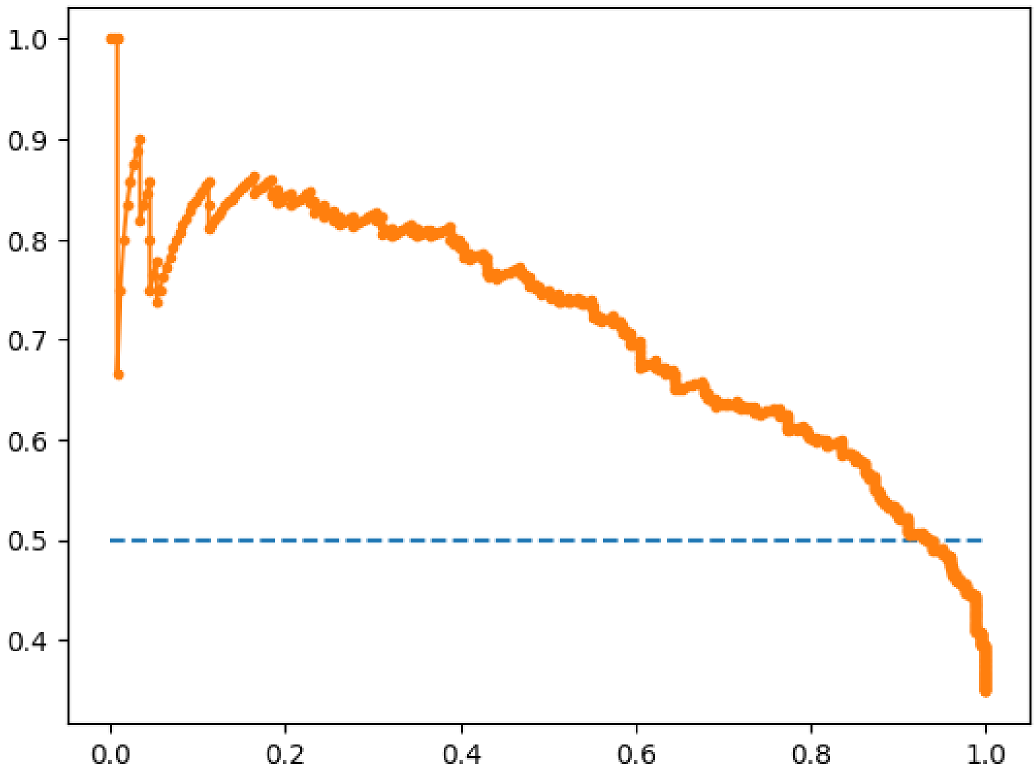
```
AUC: 0.843
True Positive Rate - [0.         0.07835821 0.26492537 0.45522388
0.62686567 0.76865672
 0.9141791  0.98880597 1.        ], False Positive Rate - [0.    0.002
0.016 0.056 0.136 0.276 0.442 0.662 1.   ] Thresholds - [2.        1.
0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

```
Text(0, 0.5, 'True Positive Rate')
```




```
#Precision Recall Curve for Logistic Regression

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.630 auc=0.713 ap=0.715

[<matplotlib.lines.Line2D at 0x1ae2dd80ac0>]
```

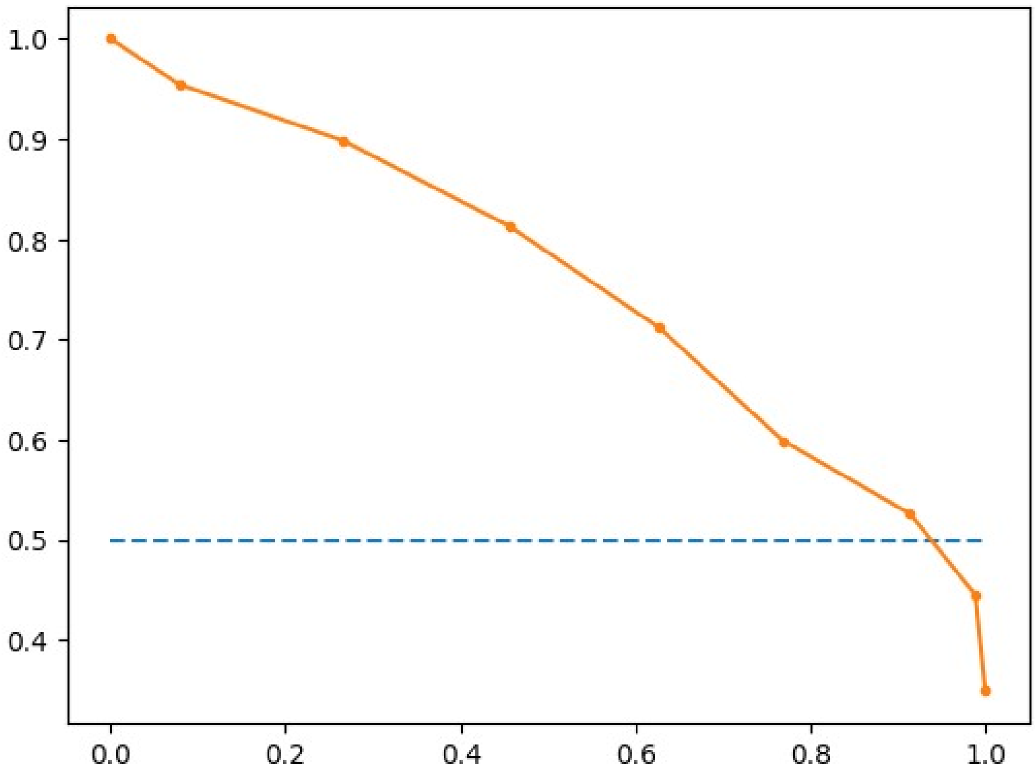



```python
#Precision Recall Curve for KNN

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
```

```
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.667 auc=0.759 ap=0.718
```

```
[<matplotlib.lines.Line2D at 0x1ae2ddfbe80>]
```

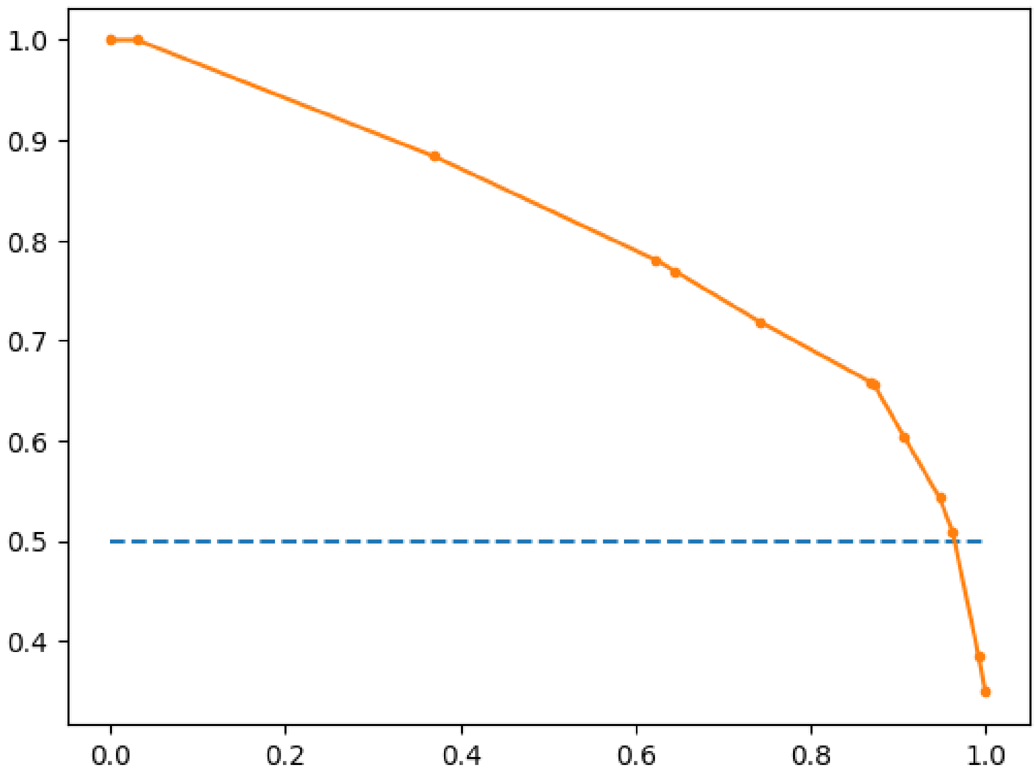



```
#Precision Recall Curve for Decission Tree Classifier

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model3.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
```

```
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.693 auc=0.809 ap=0.765
```

```
[<matplotlib.lines.Line2D at 0x1ae2de7b6a0>]
```




```
#Precision Recall Curve for Random Forest

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model4.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model4.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
```

```python
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.926 auc=0.968 ap=0.960
```

```
[<matplotlib.lines.Line2D at 0x1ae2deb27c0>]
```