1. Problem:

In [174]:

```
import numpy as np
import pandas as pd
import sqlalchemy as sa

from statsmodels.stats.outliers_influence import varia
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,

from sklearn.metrics import accuracy_score,f1_score,re
from sklearn.metrics import classification_report, cor
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import RandomOverSampler

from summarytools import dfSummary
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import pickle

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

2. Data Gathering

In [ ]:

```


This text was recognized by the built-in Ocrad engine.
```

In [22]:

```
# connecting mySQL database to Jupyter Notebook for do
con = sa.create_engine("mysql+pymysql://root:@Localhos
con
```

Out[22]:

```
Engine(mysql+pymysql://root:***@Localhost:3
306/diabetes_db)
```

In [23]:

```
df = pd.read_sql_table('diabetes',con)
df
```

Out[23]:

	Glucose	BloodPressure	SkinThickness	Insulin
0	148	50	35	0
1	85	66	29	0
2	183	64	0	0
3	150	66	23	94
4	150	40	35	168
...	...	...	...	...
763	101	76	48	180
764	122	70	27	0
765	121	72	23	112
766	126	60	0	0
767	93	70	31	0

768 rows × 8 columns

In [24]:

```
df.head()
```

Out[24]:

	Glucose	BloodPressure	SkinThickness	Insulin	BM
0	148	50	35	0	33.
1	85	66	29	0	26.
2	183	64	0	0	23.
3	150	66	23	94	28.
4	150	40	35	168	43.

3. Exploratory Data Analysis

In [25]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column                    Non-Null Cou
nt  Dtype
---  ------                    ------------
--  -----
 0   Glucose                   768 non-null
int64
 1   BloodPressure             768 non-null
int64
 2   SkinThickness             768 non-null
int64
 3   Insulin                   768 non-null
int64
 4   BMI                       768 non-null
float64
 5   DiabetesPedigreeFunction  768 non-null
float64
 6   Age                       768 non-null
int64
 7   Outcome                   768 non-null
int64
dtypes: float64(2), int64(6)
memory usage: 48.1 KB
```

In [26]:

```
dfSummary(df)
```

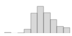
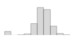
Out[26]:

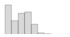
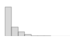
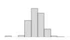
Data Frame Summary

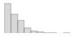
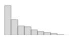
—

Dimensions: 768 x 8

Duplicates: 0

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph
1	**Glucose** [int64]	Mean (sd) : 121.1 (31.8) min < med < max: 0.0 < 117.0 < 199.0 IQR (CV) : 43.0 (3.8)	136 distinct values	
2	**BloodPressure** [int64]	Mean (sd) : 69.1 (19.4) min < med < max: 0.0 < 72.0 < 122.0 IQR (CV) : 18.0 (3.6)	47 distinct values	

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph
3	**SkinThickness** [int64]	Mean (sd) : 20.5 (16.0) min < med < max: 0.0 < 23.0 < 99.0 IQR (CV) : 32.0 (1.3)	51 distinct values	
4	**Insulin** [int64]	Mean (sd) : 79.8 (115.2) min < med < max: 0.0 < 30.5 < 846.0 IQR (CV) : 127.2 (0.7)	186 distinct values	
5	**BMI** [float64]	Mean (sd) : 32.0 (7.9) min < med < max: 0.0 < 32.0 < 67.1 IQR (CV) : 9.3 (4.1)	248 distinct values	

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph
6	**DiabetesPedigreeFunction** [float64]	Mean (sd) : 0.5 (0.3) min < med < max: 0.1 < 0.4 < 2.4 IQR (CV) : 0.4 (1.4)	517 distinct values	
7	**Age** [int64]	Mean (sd) : 33.2 (11.8) min < med < max: 21.0 < 29.0 < 81.0 IQR (CV) : 17.0 (2.8)	52 distinct values	
8	**Outcome** [int64]	Mean (sd) : 0.3 (0.5) min < med < max: 0.0 < 0.0 < 1.0 IQR (CV) : 1.0 (0.7)	2 distinct values	

In [27]:

```
# x = df.drop('Outcome',axis=1)
# y = df['Outcome']

# x_train,x_test, y_train,y_test = train_test_split(x,
```

In [28]:

```
# x_train
```

1. Glucose

In [29]:

```
df['Glucose'].head()
```

Out[29]:

```
0    148
1     85
2    183
3    150
4    150
Name: Glucose, dtype: int64
```

In [30]:

```
df['Glucose'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: Glucose
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [31]:

```
df['Glucose'].isna().sum()
```

Out[31]:

```
0
```

In [32]:

```
df['Glucose'].value_counts()
```

Out[32]:

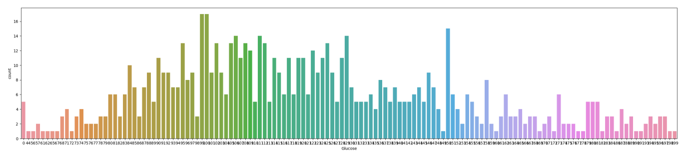
```
100    17
99     17
150    15
106    14
129    14
       ..
44      1
177     1
191     1
61      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

In [40]:

```
plt.figure(figsize= (30,6))
sns.countplot(x=df["Glucose"])
```

Out[40]:

```
<AxesSubplot:xlabel='Glucose', ylabel='coun
t'>
```

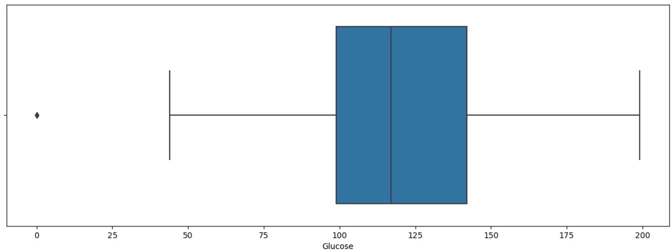


In [41]:

```
plt.figure(figsize=(15,5))
sns.boxplot(df['Glucose'])
```

Out[41]:

```
<AxesSubplot:xlabel='Glucose'>
```



2. BloodPressure

In [44]:

```
df['BloodPressure'].head()
```

Out[44]:

```
0    50
1    66
2    64
3    66
4    40
Name: BloodPressure, dtype: int64
```

In [45]:

```
df['BloodPressure'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: BloodPressure
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [47]:

```
df['BloodPressure'].isna().sum()
```

Out[47]:

```
0
```

In [48]:

```
df['BloodPressure'].value_counts()
```

Out[48]:

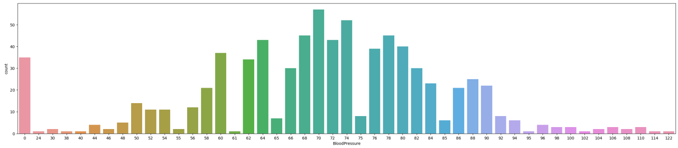
```
70      57
74      52
78      45
68      45
64      43
72      43
80      40
76      39
60      37
0       35
62      34
66      30
82      30
88      25
84      23
90      22
58      21
86      21
75       8
92       8
65       7
85       6
94       6
48       5
96       4
44       4
100      3
106      3
---      -
95       1
```

In [50]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["BloodPressure"])
```

Out[50]:

```
<AxesSubplot:xlabel='BloodPressure', ylabel
='count'>
```

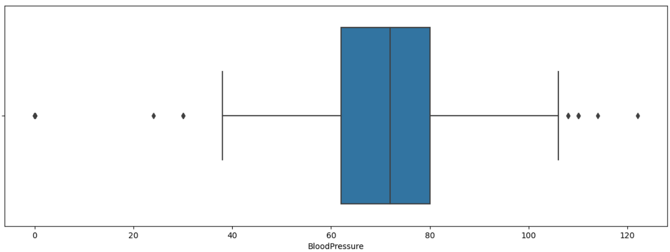


In [51]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["BloodPressure"])
```

Out[51]:

```
<AxesSubplot:xlabel='BloodPressure'>
```



```
102     1
61      1
24      1
38      1
40      1
114     1
Name: BloodPressure, dtype: int64
```

3. BloodPressure

In [53]:

```
df['SkinThickness'].head()
```

Out[53]:

```
0    35
1    29
2     0
3    23
4    35
Name: SkinThickness, dtype: int64
```

In [54]:

```
df['SkinThickness'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: SkinThickness
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [56]:

```
df['SkinThickness'].isna().sum()
```

Out[56]:

```
0
```

In [55]:

```
df['SkinThickness'].value_counts()
```

Out[55]:

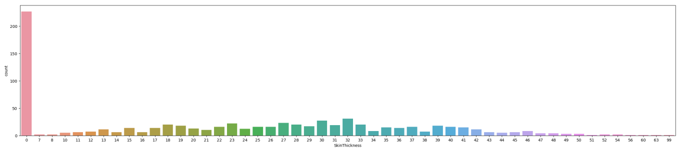
```
0     227
32     31
30     27
27     23
23     22
33     20
28     20
18     20
31     19
19     18
39     18
29     17
40     16
25     16
26     16
22     16
37     16
41     15
35     15
36     14
15     14
17     14
```

In [57]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["SkinThickness"])
```

Out[57]:

<AxesSubplot:xlabel='SkinThickness', ylabel='count'>



```
21     10
46      8
34      8
12      7
38      7
11      6
43      6
16      6
45      6
14      6
44      5
10      5
48      4
47      4
49      3
```

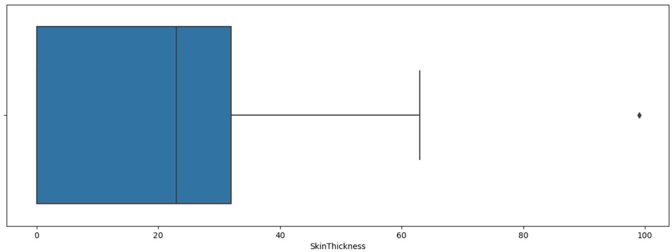
```
50      3
8       2
7       2
52      2
54      2
63      1
60      1
56      1
51      1
```

In [58]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["SkinThickness"])
```

Out[58]:

```
&lt;AxesSubplot:xlabel='SkinThickness'&gt;
```



4. Insulin

In [59]:

```
df['Insulin'].head()
```

Out[59]:

```
0      0
1      0
2      0
3     94
4    168
Name: Insulin, dtype: int64
```

In [60]:

```
df['Insulin'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: Insulin
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [61]:

```
df['Insulin'].isna().sum()
```

Out[61]:

```
0
```

In [62]:

```
df['Insulin'].value_counts()
```

Out[62]:

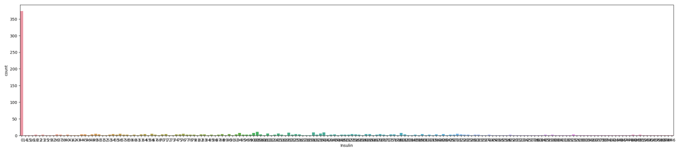
```
0      374
105     11
130      9
140      9
120      8
      ...
73       1
171      1
255      1
52       1
112      1
Name: Insulin, Length: 186, dtype: int64
```

In [63]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["Insulin"])
```

Out[63]:

```
<AxesSubplot:xlabel='Insulin', ylabel='coun
t'>
```

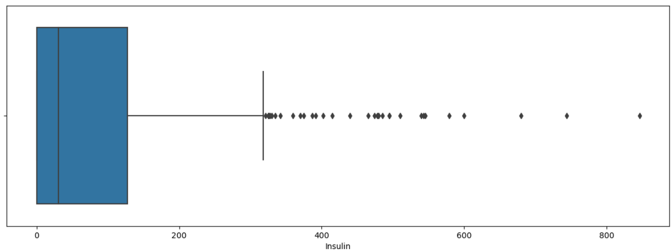


In [64]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Insulin"])
```

Out[64]:

```
<AxesSubplot:xlabel='Insulin'>
```



5. BMI

In [65]:

```
df['BMI'].head()
```

Out[65]:

```
0    33.6
1    26.6
2    23.3
3    28.1
4    43.1
Name: BMI, dtype: float64
```

In [66]:

```
df['BMI'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: BMI
Non-Null Count  Dtype
--------------  -----
768 non-null    float64
dtypes: float64(1)
memory usage: 6.1 KB
```

In [67]:

```
df['BMI'].isna().sum()
```

Out[67]:

```
0
```

In [68]:

```
df['BMI'].value_counts()
```

Out[68]:

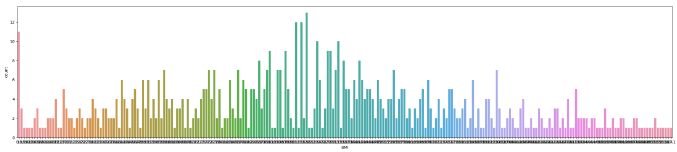
```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
        ..
36.7     1
41.8     1
42.6     1
42.8     1
46.3     1
Name: BMI, Length: 248, dtype: int64
```

In [70]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["BMI"])
```

Out[70]:

```
<AxesSubplot:xlabel='BMI', ylabel='count'>
```

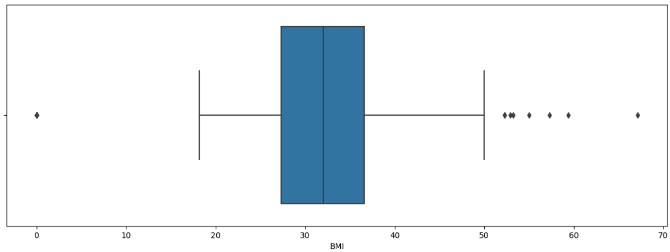


In [71]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["BMI"])
```

Out[71]:

```
<AxesSubplot:xlabel='BMI'>
```



6. DiabetesPedigreeFunction

In [72]:

```
df['DiabetesPedigreeFunction'].head()
```

Out[72]:

```
0    0.627
1    0.351
2    0.672
3    0.167
4    2.288
Name: DiabetesPedigreeFunction, dtype: floa
t64
```

In [73]:

```
df['DiabetesPedigreeFunction'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: DiabetesPedigreeFunction
Non-Null Count  Dtype  
--------------  -----  
768 non-null    float64
dtypes: float64(1)
memory usage: 6.1 KB
```

In [74]:

```
df['DiabetesPedigreeFunction'].isna().sum()
```

Out[74]:

```
0
```

In [75]:

```
df['DiabetesPedigreeFunction'].value_counts()
```

Out[75]:

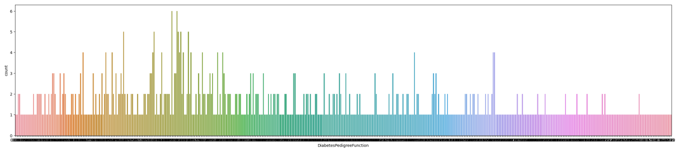
```
0.258    6
0.254    6
0.268    5
0.207    5
0.261    5
        ..
1.353    1
0.655    1
0.092    1
0.926    1
0.171    1
Name: DiabetesPedigreeFunction, Length: 51
7, dtype: int64
```

In [78]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["DiabetesPedigreeFunction"])
```

Out[78]:

```
<AxesSubplot:xlabel='DiabetesPedigreeFuncti
on', ylabel='count'>
```

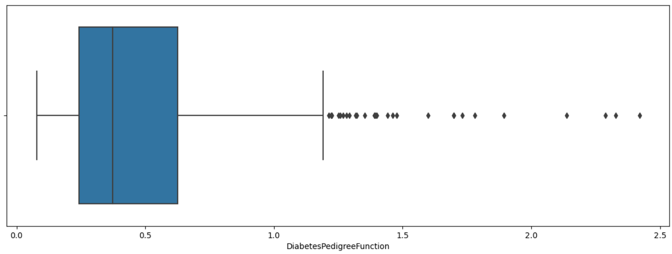


In [77]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["DiabetesPedigreeFunction"])
```

Out[77]:

```
<AxesSubplot:xlabel='DiabetesPedigreeFuncti
on'>
```



7. Age

In [80]:

```
df['Age'].head()
```

Out[80]:

```
0    50
1    31
2    52
3    21
4    33
Name: Age, dtype: int64
```

In [81]:

```
df['Age'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: Age
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [82]:

```
df['Age'].isna().sum()
```

Out[82]:

```
0
```

In [83]:

```
df['Age'].value_counts()
```

Out[83]:

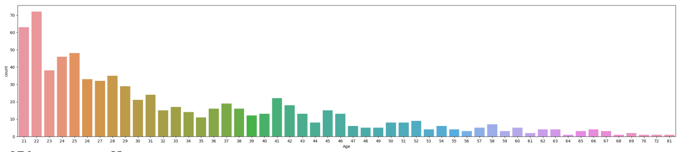
```
22     72
21     63
25     48
24     46
23     38
28     35
26     33
27     32
29     29
31     24
41     22
30     21
37     19
42     18
33     17
38     16
36     16
32     15
45     15
34     14
43     13
46     13
```

In [85]:

```
plt.figure(figsize= (30,6))
sns.countplot(df["Age"])
```

Out[85]:

```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



```
50      8
51      8
44      8
58      7
47      6
54      6
49      5
48      5
57      5
60      5
66      4
53      4
62      4
55      4
63      4
```

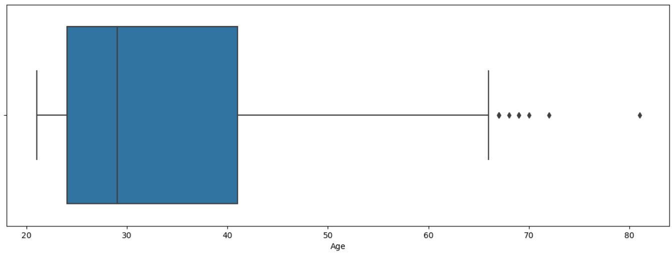
```
67      3
56      3
59      3
65      3
69      2
61      2
72      1
81      1
64      1
```

In [86]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Age"])
```

Out[86]:

```
&lt;AxesSubplot:xlabel='Age'&gt;
```



8. Outcome

In [89]:

```
df['Outcome']
```

Out[89]:

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [90]:

```
df['Outcome'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: Outcome
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [91]:

```
df['Outcome'].isna().sum()
```

Out[91]:

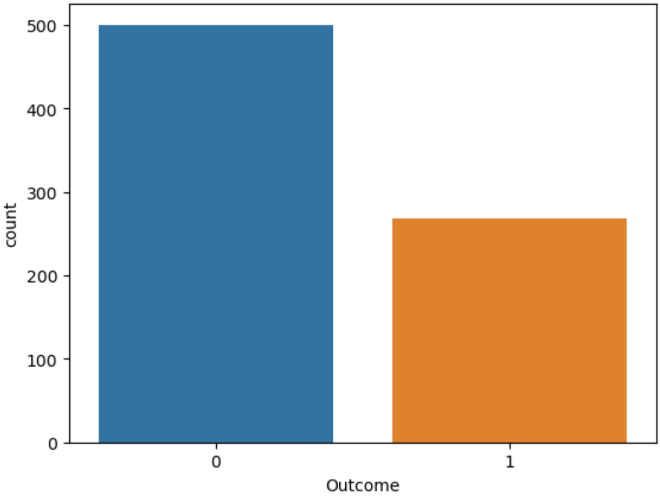
```
0
```

In [94]:

```
sns.countplot(df['Outcome'])
```

Out[94]:

```
<AxesSubplot:xlabel='Outcome', ylabel='coun
t'>
```

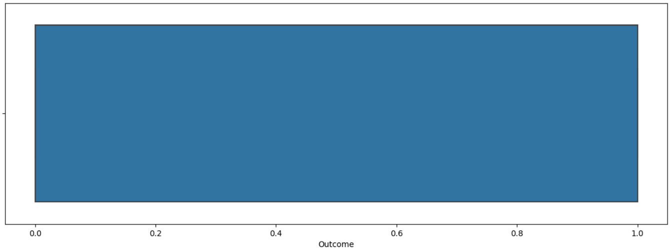


In [95]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Outcome"])
```

Out[95]:

<AxesSubplot:xlabel='Outcome'>



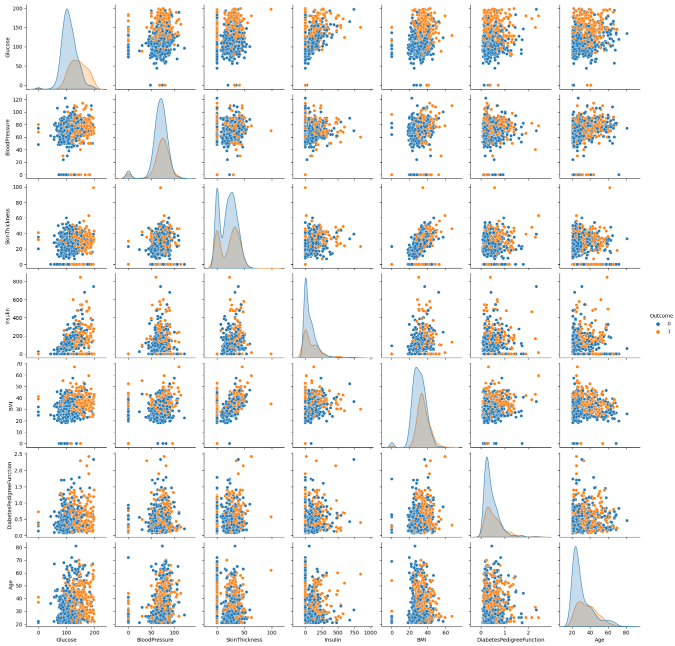
4. Feature Engineering

In [97]:

```
sns.pairplot(df, hue = 'Outcome')
```

Out[97]:

<seaborn.axisgrid.PairGrid at 0x198b6bdba30
>



4.1 zero value Imputation

In [113]:

```
num= df[df["SkinThickness"]==0]
num1= df[df["BloodPressure"]==0]
num2= df[df["Glucose"]==0]
num3= df[df["Insulin"]==0]
num4= df[df["BMI"]==0]
num.shape,num1.shape,num2.shape,num3.shape,num4.shape
```

Out[113]:

```
((227, 8), (35, 8), (5, 8), (374, 8), (11,
8))
```

In [114]:

```
df[['Glucose', 'BloodPressure', 'BMI','Insulin','Skin
```

In [115]:

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

Out[115]:

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

In [117]:

```
df[['Glucose', 'BloodPressure', 'BMI','Insulin','Skin
```

In [119]:

```
num= df[df["SkinThickness"]==0]
num1= df[df["BloodPressure"]==0]
num2= df[df["Glucose"]==0]
num3= df[df["Insulin"]==0]
num4= df[df["BMI"]==0]
num.shape,num1.shape,num2.shape,num3.shape,num4.shape
```

Out[119]:

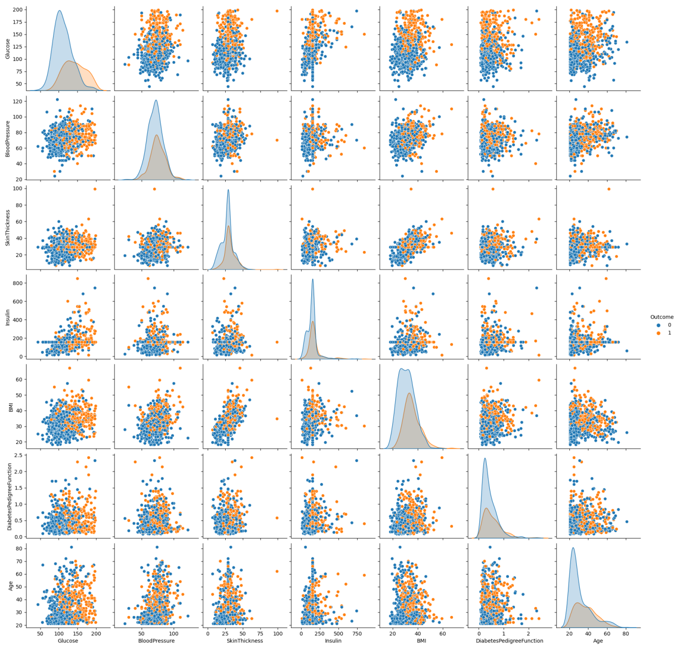
```
((0, 8), (0, 8), (0, 8), (0, 8), (0, 8))
```

In [132]:

```
sns.pairplot(df, hue = 'Outcome')
```

Out[132]:

```
<seaborn.axisgrid.PairGrid at 0x198bbefec40
>
```



4.2 Outlier Imputation

In [134]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column                    Non-Null Cou
nt  Dtype
---  ------                    ------------
--  -----
 0   Glucose                   768 non-null
float64
 1   BloodPressure             768 non-null
float64
 2   SkinThickness             768 non-null
float64
 3   Insulin                   768 non-null
float64
 4   BMI                       768 non-null
float64
 5   DiabetesPedigreeFunction  768 non-null
float64
 6   Age                       768 non-null
int64
 7   Outcome                   768 non-null
int64
dtypes: float64(6), int64(2)
memory usage: 48.1 KB
```

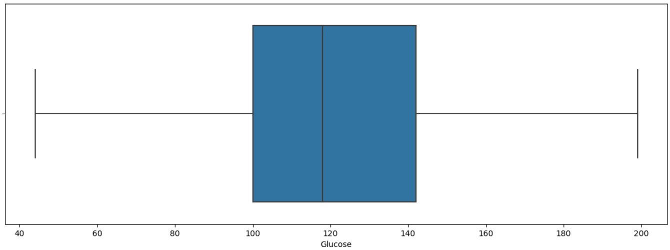
1.Glucose

In [136]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Glucose"])
```

Out[136]:

```
<AxesSubplot:xlabel='Glucose'>
```



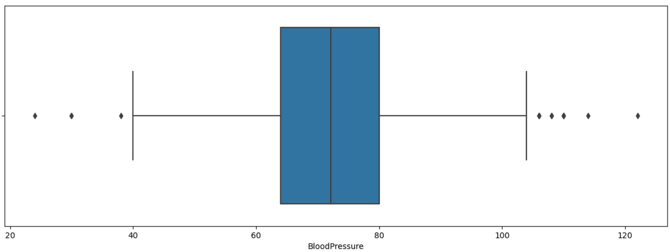
2. BloodPressure

In [137]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["BloodPressure"])
```

Out[137]:

```
<AxesSubplot:xlabel='BloodPressure'>
```



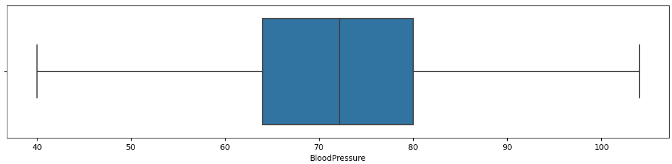
In [181]:

```
# Removing outliers with IQR method

q1 = df['BloodPressure'].quantile(0.25)
q2 = df['BloodPressure'].quantile(0.50)
q3 = df['BloodPressure'].quantile(0.75)
median = df['BloodPressure'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['BloodPressure'].loc[(df['BloodPressure'] > upp
print("upper_tail outliers:\n ", a)
b= df['BloodPressure'].loc[(df['BloodPressure'] < lowe
print("lower_tail outliers: \n", b)
df.loc[(df['BloodPressure']> upper_tail),'BloodPressu
df.loc[(df['BloodPressure']< lower_tail),'BloodPressu

plt.figure(figsize=(15,3))
sns.boxplot(df['BloodPressure'])
```

```
Q1 : 64.0
Q2 : 72.18758526603001
Q3 : 80.0
Median : 72.18758526603001
upper_tail : 104.0
lower_tail : 40.0
upper_tail outliers:
  Series([], Name: BloodPressure, dtype: fl
oat64)
lower_tail outliers:
 18     30.0
125    30.0
597    24.0
599    38.0
Name: BloodPressure, dtype: float64
```

Out[181]:

```
<AxesSubplot:xlabel='BloodPressure'>
```



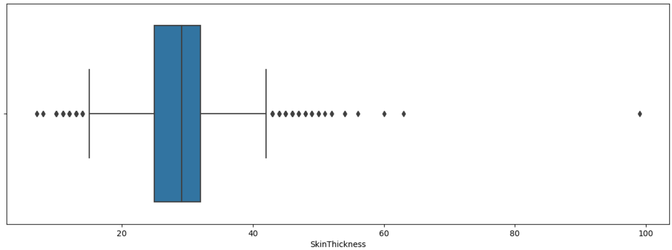
3. SkinThickness

In [138]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["SkinThickness"])
```

Out[138]:

<AxesSubplot:xlabel='SkinThickness'>



In [184]:

```
# Removing outliers with IQR method

q1 = df['SkinThickness'].quantile(0.25)
q2 = df['SkinThickness'].quantile(0.50)
q3 = df['SkinThickness'].quantile(0.75)
median = df['SkinThickness'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['SkinThickness'].loc[(df['SkinThickness'] > upp
print("upper_tail outliers:\n ", a)
b= df['SkinThickness'].loc[(df['SkinThickness'] < lowe
print("lower_tail outliers: \n", b)
df.loc[(df['SkinThickness']> upper_tail),'SkinThicknes
df.loc[(df['SkinThickness']< lower_tail),'SkinThicknes

```

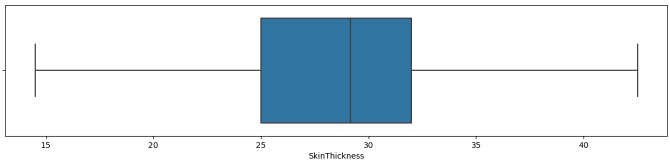
```
Q1 : 25.0
Q2 : 29.153419593345657
Q3 : 32.0
Median : 29.153419593345657
upper_tail : 42.5
lower_tail : 14.5
upper_tail outliers:
  Series([], Name: SkinThickness, dtype: fl
oat64)
lower_tail outliers:
 Series([], Name: SkinThickness, dtype: flo
at64)
```

In [185]:

```
plt.figure(figsize=(15,3))
sns.boxplot(df['SkinThickness'])
```

Out[185]:

```
<AxesSubplot:xlabel='SkinThickness'>
```



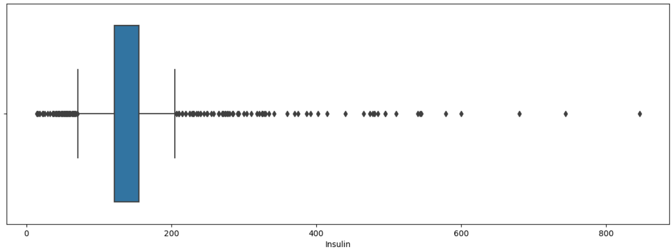
4. Insulin

In [139]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Insulin"])
```

Out[139]:

```
<AxesSubplot:xlabel='Insulin'>
```



In [187]:

```
# Removing outliers with IQR method

q1 = df['Insulin'].quantile(0.25)
q2 = df['Insulin'].quantile(0.50)
q3 = df['Insulin'].quantile(0.75)
median = df['Insulin'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['Insulin'].loc[(df['Insulin'] > upper_tail)]
print("upper_tail outliers:\n ", a)
b= df['Insulin'].loc[(df['Insulin'] < lower_tail)]
print("lower_tail outliers: \n", b)
df.loc[(df['Insulin']> upper_tail),'Insulin']= upper_t
df.loc[(df['Insulin']< lower_tail),'Insulin']= lower_t

```

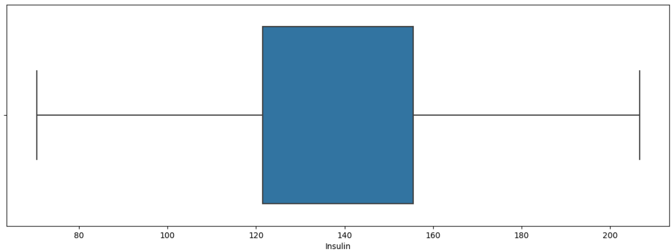
```
Q1 : 121.5
Q2 : 155.5482233502538
Q3 : 155.5482233502538
Median : 155.5482233502538
upper_tail : 206.62055837563452
lower_tail : 70.42766497461929
upper_tail outliers:
  8      543.0
13     846.0
16     230.0
20     235.0
31     245.0
       ...
707    335.0
710    387.0
713    291.0
715    392.0
753    510.0
Name: Insulin, Length: 82, dtype: float64
lower_tail outliers:
 32     54.0
40     70.0
51     36.0
52     23.0
68     38.0
       ...
672    49.0
680    45.0
711    22.0
747    57.0
760    16.0
Name: Insulin, Length: 82, dtype: float64
```

In [188]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Insulin"])
```

Out[188]:

<AxesSubplot:xlabel='Insulin'>



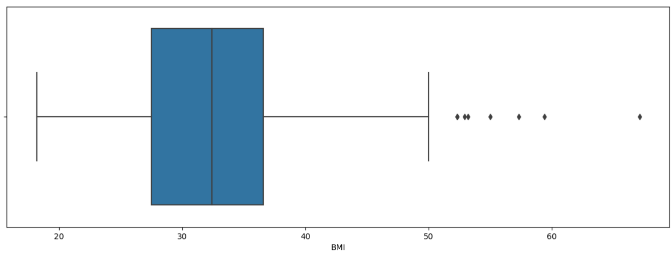
5. BMI

In [141]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["BMI"])
```

Out[141]:

<AxesSubplot:xlabel='BMI'>



In [189]:

```
# Removing outliers with IQR method

q1 = df['BMI'].quantile(0.25)
q2 = df['BMI'].quantile(0.50)
q3 = df['BMI'].quantile(0.75)
median = df['BMI'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['BMI'].loc[(df['BMI'] > upper_tail)]
print("upper_tail outliers:\n ", a)
b= df['BMI'].loc[(df['BMI'] < lower_tail)]
print("lower_tail outliers: \n", b)
df.loc[(df['BMI']> upper_tail),'BMI']= upper_tail
df.loc[(df['BMI']< lower_tail),'BMI']= lower_tail

```

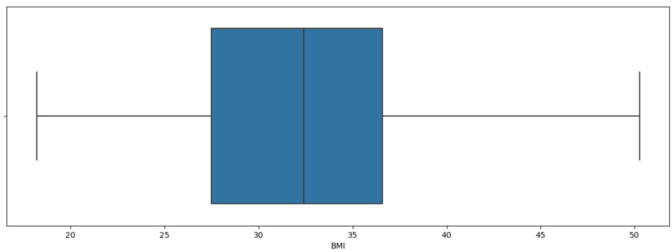
```
Q1 : 27.5
Q2 : 32.4
Q3 : 36.6
Median : 32.4
upper_tail : 50.25
lower_tail : 13.849999999999998
upper_tail outliers:
  120    53.2
125    55.0
177    67.1
193    52.3
247    52.3
303    52.9
445    59.4
673    57.3
Name: BMI, dtype: float64
lower_tail outliers:
 Series([], Name: BMI, dtype: float64)
```

In [190]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["BMI"])
```

Out[190]:

```
<AxesSubplot:xlabel='BMI'>
```



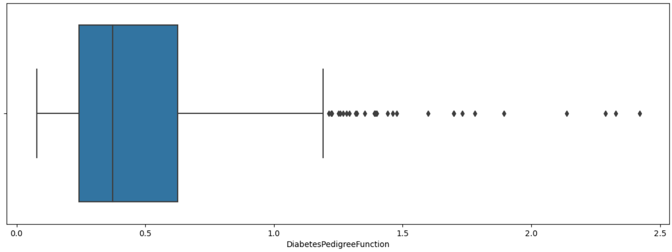
6. DiabetesPedigreeFunction

In [142]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["DiabetesPedigreeFunction"])
```

Out[142]:

```
<AxesSubplot:xlabel='DiabetesPedigreeFuncti
on'>
```



In [191]:

```
# Removing outliers with IQR method

q1 = df['DiabetesPedigreeFunction'].quantile(0.25)
q2 = df['DiabetesPedigreeFunction'].quantile(0.50)
q3 = df['DiabetesPedigreeFunction'].quantile(0.75)
median = df['DiabetesPedigreeFunction'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['DiabetesPedigreeFunction'].loc[(df['DiabetesPe
print("upper_tail outliers:\n ", a)
b= df['DiabetesPedigreeFunction'].loc[(df['DiabetesPed
print("lower_tail outliers: \n", b)
df.loc[(df['DiabetesPedigreeFunction']> upper_tail),'D
df.loc[(df['DiabetesPedigreeFunction']< lower_tail),'D

```

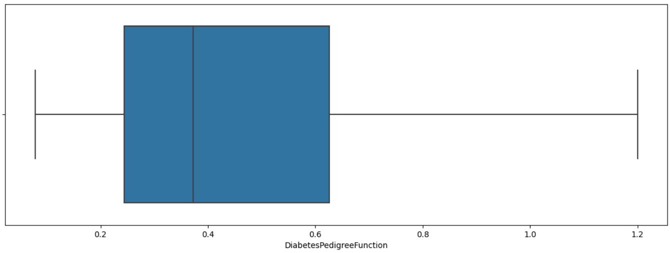
```
Q1 : 0.24375
Q2 : 0.3725
Q3 : 0.62625
Median : 0.3725
upper_tail : 1.2
lower_tail : -0.32999999999999996
upper_tail outliers:
  4      2.288
12     1.441
39     1.390
45     1.893
58     1.781
100    1.222
147    1.400
187    1.321
218    1.224
228    2.329
243    1.318
245    1.213
259    1.353
292    1.224
308    1.391
330    1.476
370    2.137
371    1.731
383    1.268
395    1.600
445    2.420
534    1.251
593    1.699
606    1.258
618    1.282
621    1.698
622    1.461
659    1.292
661    1.394
Name: DiabetesPedigreeFunction, dtype: floa
t64
lower_tail outliers:
 Series([], Name: DiabetesPedigreeFunction,
dtype: float64)
```

In [192]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["DiabetesPedigreeFunction"])
```

Out[192]:

```
<AxesSubplot:xlabel='DiabetesPedigreeFuncti
on'>
```



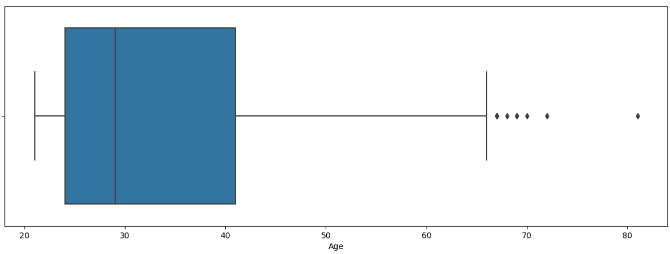
7. Age

In [143]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Age"])
```

Out[143]:

```
<AxesSubplot:xlabel='Age'>
```



In [193]:

```
# Removing outliers with IQR method

q1 = df['Age'].quantile(0.25)
q2 = df['Age'].quantile(0.50)
q3 = df['Age'].quantile(0.75)
median = df['Age'].median()

iqr = q3 - q1

upper_tail = q3 + 1.5 * iqr
lower_tail = q1 - 1.5 * iqr

print("Q1 :", q1)
print("Q2 :", q2)
print("Q3 :", q3)
print("Median :",median)

print("upper_tail :", upper_tail)
print("lower_tail :", lower_tail)
a = df['Age'].loc[(df['Age'] > upper_tail)]
print("upper_tail outliers:\n ", a)
b= df['Age'].loc[(df['Age'] < lower_tail)]
print("lower_tail outliers: \n", b)
df.loc[(df['Age']> upper_tail),'Age']= upper_tail
df.loc[(df['Age']< lower_tail),'Age']= lower_tail

```

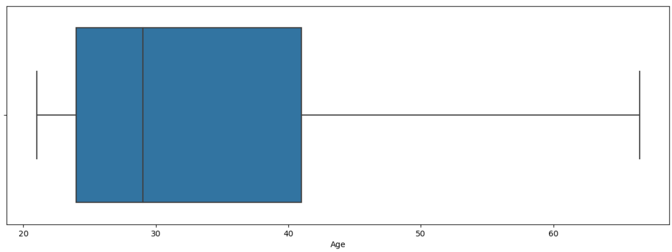
```
Q1 : 24.0
Q2 : 29.0
Q3 : 41.0
Median : 29.0
upper_tail : 66.5
lower_tail : -1.5
upper_tail outliers:
  123    69
363    67
453    72
459    81
489    67
537    67
666    70
674    68
684    69
Name: Age, dtype: int64
lower_tail outliers:
 Series([], Name: Age, dtype: int64)
```

In [194]:

```
plt.figure(figsize= (15,5))
sns.boxplot(df["Age"])
```

Out[194]:

```
<AxesSubplot:xlabel='Age'>
```



5. Feature Selection

5.1 Assumption

In [144]:

```
df.corr()
```

Out[144]:

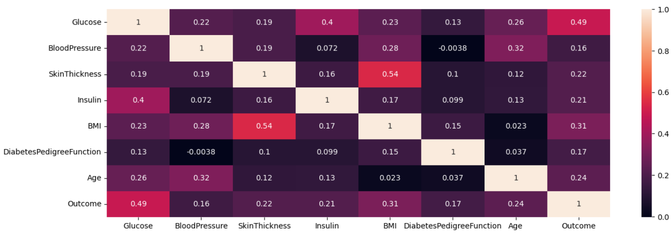
	Glucose	BloodPressure	Sk
Glucose	1.000000	0.215938	
BloodPressure	0.215938	1.000000	
SkinThickness	0.192013	0.190815	
Insulin	0.395345	0.072356	
BMI	0.232153	0.280253	
DiabetesPedigreeFunction	0.134736	-0.003836	
Age	0.259389	0.319440	
Outcome	0.489601	0.162449	

In [145]:

```
plt.figure(figsize=(15,5))
sns.heatmap(df.corr(), annot=True)
```

Out[145]:

```
<AxesSubplot:>
```



5.2. No Multicolinearity

In [148]:

```
df1 = df.drop('Outcome', axis= 1)
df1.head()
```

Out[148]:

	Glucose	BloodPressure	SkinThickness	Insulin
0	148.0	50.0	35.00000	155.548223
1	85.0	66.0	29.00000	155.548223
2	183.0	64.0	29.15342	155.548223
3	150.0	66.0	23.00000	94.000000
4	150.0	40.0	35.00000	168.000000

In [151]:

```
vif_data = pd.DataFrame()
vif_data["feature"] = df1.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(df1.value
 for i in range(len(df1.columns))]

print(vif_data)
```

```
                    feature        VIF
0                   Glucose  21.347959
1             BloodPressure  31.335650
2             SkinThickness  17.360263
3                   Insulin   5.232267
4                       BMI  33.628317
5  DiabetesPedigreeFunction   3.131796
6                       Age  10.680453
```

6. Model Building

6.1. sampling

In [ ]:

```

```

6.2. Splitting the data into Training data and Testing data

In [227]:

```
# regular Model
# independent variable(x)
x = df.drop('Outcome', axis= 1)
# dependent variable(y)
y = df['Outcome']
x_train, x_test, y_train, y_test = train_test_split(x,
```

In [228]:

```
print("shape of x_train:",x_train.shape)
print("shape of y_train:",y_train.shape)
print("shape of x_test:",x_test.shape)
print("shape of y_test:",y_test.shape)
```

```
shape of x_train: (537, 7)
shape of y_train: (537,)
shape of x_test: (231, 7)
shape of y_test: (231,)
```

6.3 Model Training

Logistic Regression

In [229]:

```
logistic_model = LogisticRegression()
logistic_model.fit(x_train,y_train)
```

Out[229]:

```
LogisticRegression()
```

In [230]:

```
logistic_pred =logistic_model.predict(x_test)
logistic_pred[:5]
```

Out[230]:

```
array([0, 0, 1, 0, 0], dtype=int64)
```

6.4 Model Evaluation

Fucntion for test and train

In [231]:

```
# Testing Data
def testing_evalution(model,x,y):
    prediction = model.predict(x)

    cnf_matrix = confusion_matrix(y, prediction)
    print("Confusion Matrix:\n", cnf_matrix)
    print("*"*45)

    accuracy = accuracy_score(y, prediction)
    print('Accuracy \n',accuracy)
    print("*"*45)

    clf_report= classification_report(y, prediction)
    print("Classification Report\n", clf_report)
```

In [232]:

```
# Training Data
def training_evalution(model,x,y):
    prediction = model.predict(x)

    cnf_matrix = confusion_matrix(y, prediction)
    print("Confusion Matrix:\n", cnf_matrix)
    print("*"*45)

    accuracy = accuracy_score(y, prediction)
    print('Accuracy \n',accuracy)
    print("*"*45)

    clf_report= classification_report(y, prediction)
    print("Classification Report\n", clf_report)
```

regular Model

In [233]:

```
# Testing Data
testing_evalution(logistic_model,x_test,y_test)
```

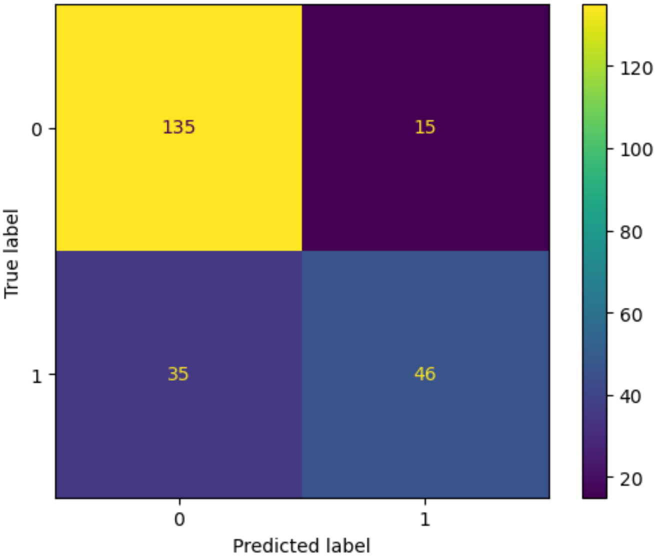
```
Confusion Matrix:
 [[135  15]
 [ 35  46]]
******************************************
**
Accuracy
 0.7835497835497836
******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.79      0.90      0.84
150
           1       0.75      0.57      0.65
81

    accuracy                           0.78
231
   macro avg       0.77      0.73      0.75
231
weighted avg       0.78      0.78      0.78
231
```

In [234]:

```
plot_confusion_matrix(logistic_model,x_test,y_test)
```

Out[234]:

```
<sklearn.metrics._plot.confusion_matrix.Con
fusionMatrixDisplay at 0x198c466ae50>
```



In [235]:

```
# Training Data
training_evalution(logistic_model,x_train,y_train)
```

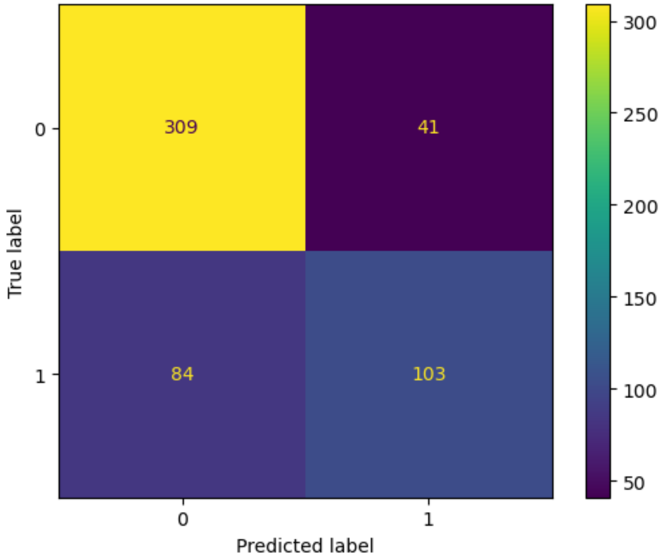
```
Confusion Matrix:
 [[309  41]
 [ 84 103]]
*******************************************
**
Accuracy
 0.7672253258845437
*******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.79      0.88      0.83
350
           1       0.72      0.55      0.62
187

    accuracy                           0.77
537
   macro avg       0.75      0.72      0.73
537
weighted avg       0.76      0.77      0.76
537
```

In [236]:

```
plot_confusion_matrix(logistic_model,x_train,y_train)
```

Out[236]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x198c7033d60>
```



checking roc curve :

In [237]:

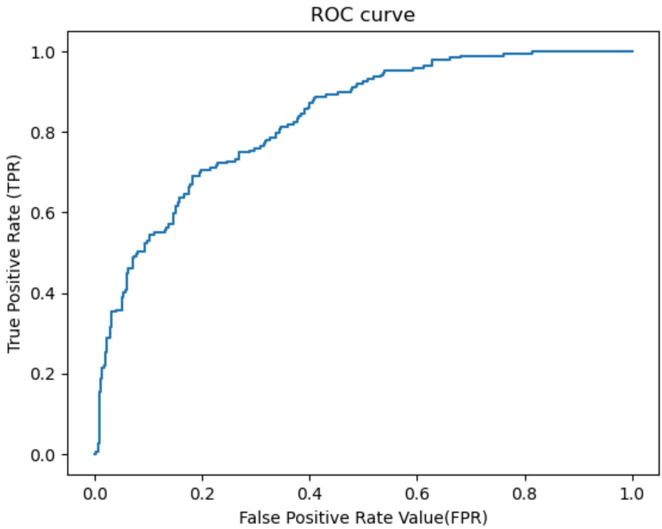
```
y_pred_proba = logistic_model.predict_proba(x_train)
y_pred_proba[:,1]
fpr,tpr,thresh = roc_curve(y_train,y_pred_proba[:,1])
```

In [238]:

```
plt.plot(fpr,tpr)
plt.xlabel("False Positive Rate Value(FPR)")
plt.ylabel("True Positive Rate (TPR)")
plt.title("ROC curve")
```

Out[238]:

```
Text(0.5, 1.0, 'ROC curve')
```



In [239]:

```
### auc value
auc_value = auc(fpr,tpr)
auc_value
```

Out[239]:

```
0.8310313216195568
```

In [240]:

```
thresh = np.arange(0,1,0.1)
info_df = pd.DataFrame()

for i in thresh:
    preds = (logistic_model.predict_proba(x_test)[:,1]

    thresh_df=pd.DataFrame(data=[accuracy_score(y_test
                                index=["Accuracy","Recall",'
    info_df = pd.concat([info_df,thresh_df],axis=1)


info_df.columns=thresh
info_df
```

Out[240]:

	0.0	0.1	0.2	0.3	
Accuracy	0.350649	0.571429	0.718615	0.774892	0.78
Recall	1.000000	1.000000	0.913580	0.765432	0.66
Precision	0.350649	0.450000	0.560606	0.652632	0.71
F1-score	0.519231	0.620690	0.694836	0.704545	0.68

Hyper-Parameter Tunning :

In [241]:

```
weights = np.linspace(0,0.99,num=100)
param_grid = {"C":np.arange(0.1,20),"penalty":['l1','"


gscv_model = GridSearchCV(logistic_model,param_grid)
gscv_model.fit(x_train,y_train)
```

Out[241]:

```
GridSearchCV(estimator=LogisticRegression
(),
             param_grid={'C': array([ 0.1,
1.1,  2.1,  3.1,  4.1,  5.1,  6.1,  7.1,
8.1,  9.1, 10.1,
       11.1, 12.1, 13.1, 14.1, 15.1, 16.1,
17.1, 18.1, 19.1]),
                         'class_weight':
[{0: 0.0, 1: 1.0}, {0: 0.01, 1: 0.99},

{0: 0.02, 1: 0.98},

{0: 0.03, 1: 0.97},

{0: 0.04, 1: 0.96},

{0: 0.05, 1: 0.95},

{0: 0.06, 1: 0.94},

{0: 0.07, 1: 0.9299999999999999},

{0: 0.08, 1: 0.92},

{0: 0.09, 1: 0.91}, {0:...: 0.9},

{0: 0.11, 1: 0.89},

{0: 0.12, 1: 0.88},

{0: 0.13, 1: 0.87},

{0: 0.14, 1: 0.86},

{0: 0.15, 1: 0.85},

{0: 0.16, 1: 0.84},

{0: 0.17, 1: 0.83},

{0: 0.18, 1: 0.8200000000000001},
```

In [242]:

```
gscv_model.best_params_
```

Out[242]:

```
{'C': 13.1, 'class_weight': {0: 0.55, 1: 0.
4499999999999996}, 'penalty': 'l2'}
```

```
{0: 0.19, 1: 0.81}, {0: 0.2, 1: 0.8},

{0: 0.21, 1: 0.79},

{0: 0.22, 1: 0.78},

{0: 0.23, 1: 0.77},

{0: 0.24, 1: 0.76},

{0: 0.25, 1: 0.75},

{0: 0.26, 1: 0.74},

{0: 0.27, 1: 0.73},

{0: 0.28, 1: 0.72},

{0: 0.29, 1: 0.71}, ...],
                            'penalty': ['l1',
'l2']})
```

In [243]:

```
ht_logistic_model = LogisticRegression(penalty='l2',C=
ht_logistic_model.fit(x_train,y_train)
```

Out[243]:

```
LogisticRegression(C=6.1, class_weight={0:
0.5, 1: 0.5})
```

In [244]:

```
# testing data :

testing_evalution(ht_logistic_model,x_test,y_test)
```

```
Confusion Matrix:
 [[134  16]
 [ 35  46]]
******************************************
**
Accuracy
 0.7792207792207793
******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.79      0.89      0.84
150
           1       0.74      0.57      0.64
81

    accuracy                           0.78
231
   macro avg       0.77      0.73      0.74
231
weighted avg       0.78      0.78      0.77
231
```

In [245]:

```
# training data

training_evalution(ht_logistic_model,x_train,y_train)
```

```
Confusion Matrix:
 [[310  40]
 [ 84 103]]
******************************************
**
Accuracy
 0.7690875232774674
******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.79      0.89      0.83
350
           1       0.72      0.55      0.62
187

    accuracy                           0.77
537
   macro avg       0.75      0.72      0.73
537
weighted avg       0.76      0.77      0.76
537
```

6.3 Model Training aftter outlier removing

In [195]:

```
logistic_model_OR = LogisticRegression()
logistic_model_OR.fit(x_train, y_train)
```

Out[195]:

```
LogisticRegression()
```

In [196]:

```
logistic_pred =logistic_model_OR.predict(x_test)
logistic_pred[:5]
```

Out[196]:

```
array([0, 0, 0, 1, 0], dtype=int64)
```

6.4 Model Evaluation

In [197]:

```
# Testing Data
testing_evalution(logistic_model_OR,x_test,y_test)
```

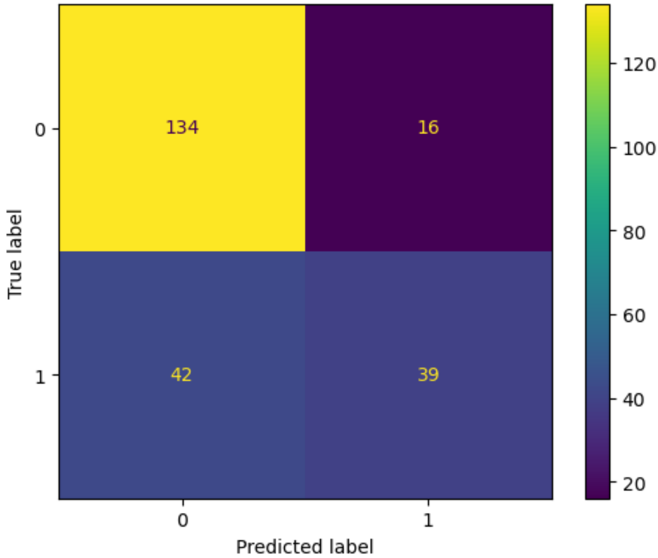
```
Confusion Matrix:
 [[134  16]
 [ 42  39]]
******************************************
**
Accuracy
 0.7489177489177489
******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.76      0.89      0.82
150
           1       0.71      0.48      0.57
81

    accuracy                           0.75
231
   macro avg       0.74      0.69      0.70
231
weighted avg       0.74      0.75      0.73
231
```

In [198]:

```
1 plot_confusion_matrix(logistic_model_OR,x_test,y_test)
```

Out[198]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x198c40af670>
```



In [199]:

```
# Training Data
training_evalution(logistic_model_OR,x_train,y_train)
```

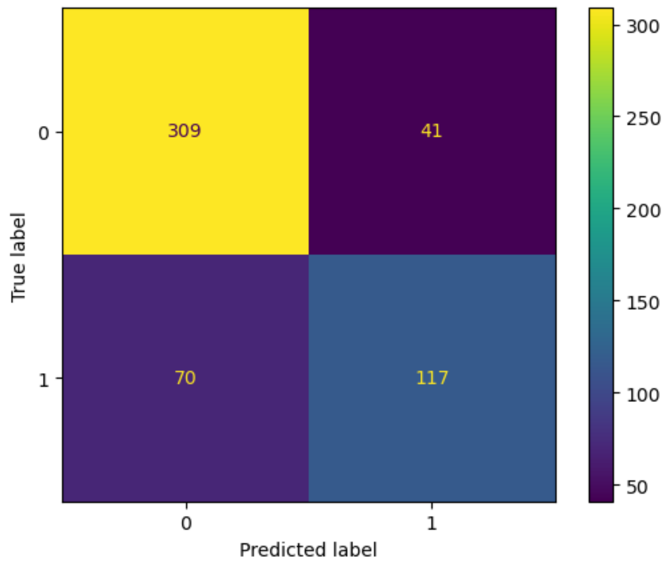
```
Confusion Matrix:
 [[309  41]
 [ 70 117]]
*******************************************
**
Accuracy
 0.7932960893854749
*******************************************
**
Classification Report
               precision    recall  f1-scor
e   support

           0       0.82      0.88      0.85
350
           1       0.74      0.63      0.68
187

    accuracy                           0.79
537
   macro avg       0.78      0.75      0.76
537
weighted avg       0.79      0.79      0.79
537
```

In [200]:

```
plot_confusion_matrix(logistic_model_OR,x_train,y_trai
```

Out[200]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x198c432d6a0>
```



Pickle File

In [246]:

```
with open('Logistic model.pkl', 'wb') as f:
    pickle.dump(logistic_model,f)
```

Single user Input Testing

In [247]:

```
x_test.iloc[35]
```

Out[247]:

```
Glucose                     141.000000
BloodPressure                72.375171
SkinThickness                29.153420
Insulin                     155.548223
BMI                          42.400000
DiabetesPedigreeFunction      0.205000
Age                          29.000000
Name: 435, dtype: float64
```

In [248]:

```
Glucose = 141.000000
BloodPressure = 72.375171
SkinThickness = 29.153420
Insulin = 155.548223
BMI = 42.400000
DiabetesPedigreeFunction = 0.205000
Age = 29.000000
```

In [249]:

```
test_array = np.array([[Glucose, BloodPressure, SkinTh
test_array
```

Out[249]:

```
array([[141.      ,  72.375171,  29.15342 ,
155.548223,  42.4     ,
          0.205    ,  29.      ]])
```

In [250]:

```
prediction = logistic_model.predict(test_array)
prediction
```

Out[250]:

```
array([1], dtype=int64)
```

In [251]:

```
if prediction==1:
    print("Yes,You are Having Diabetics")
else:
    print("No,You are not Having Diabetics")
```

```
Yes,You are Having Diabetics
```

In [ ]:

```

```