In [121]:

```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [122]:

```
diabetes=pd.read_csv("Diabetes.csv")
diabetes.head()
```

Out[122]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [123]:

```
diabetes.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 [124]:

```
diabetes.describe()
```

Out[124]:

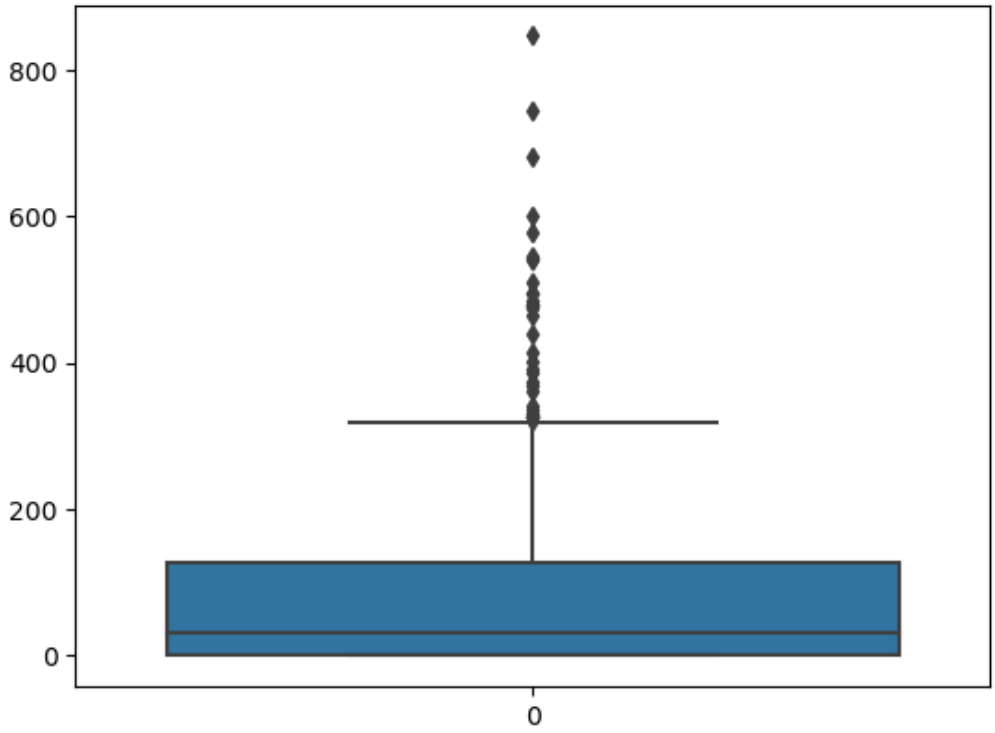
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.952218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

In [125]:

```
sns.boxplot(diabetes['Insulin'])
```

Out[125]:

```
<Axes: >
```

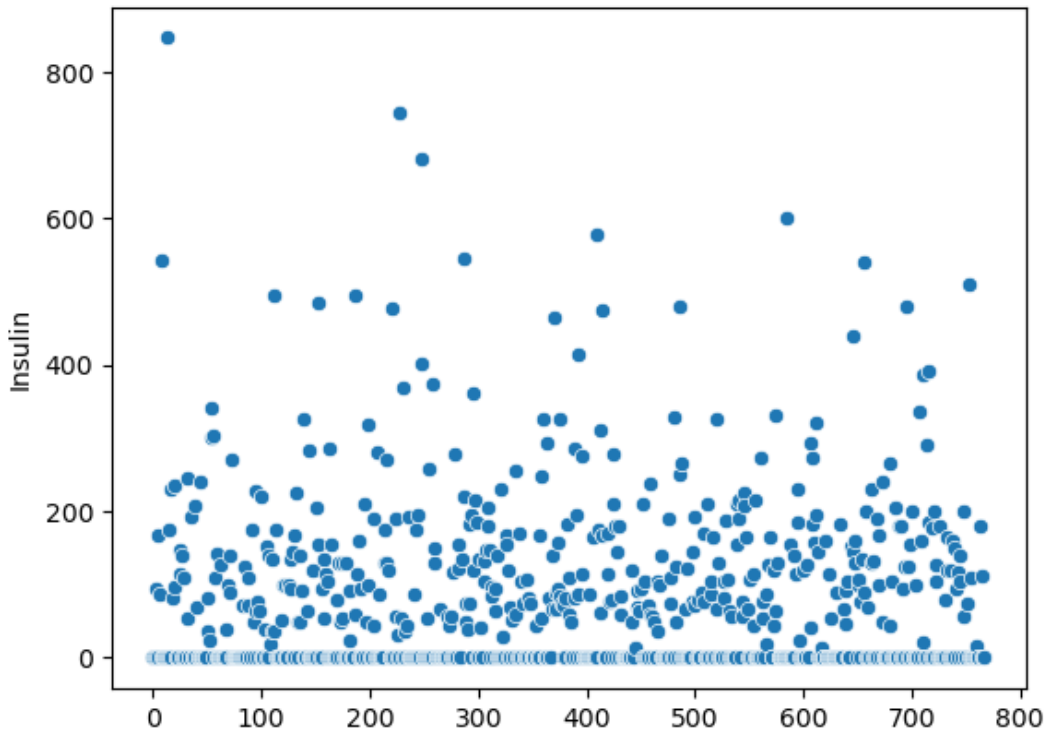


In [126]:

```
sns.scatterplot(diabetes['Insulin'])
```

Out[126]:

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



In [127]:

```
np.percentile(diabetes.Insulin,[99])
```

Out[127]:

```
array([519.9])
```

In [128]:

```
uv=np.percentile(diabetes.Insulin,[99])[0]
```

In [129]:

```
diabetes[(diabetes.Insulin>uv)]
```

Out[129]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
8	2	197	70	45	543	30.5	0.158	53	1
13	1	189	60	23	846	30.1	0.398	59	1
228	4	197	70	39	744	36.7	2.329	31	0
247	0	165	90	33	680	52.3	0.427	23	0
286	5	155	84	44	545	38.7	0.619	34	0
409	1	172	68	49	579	42.4	0.702	28	1
584	8	124	76	24	600	28.7	0.687	52	1
655	2	155	52	27	540	38.7	0.240	25	1

In [ ]:

```

```

In [130]:

```
diabetes.Insulin[(diabetes.Insulin>uv)]=uv
```

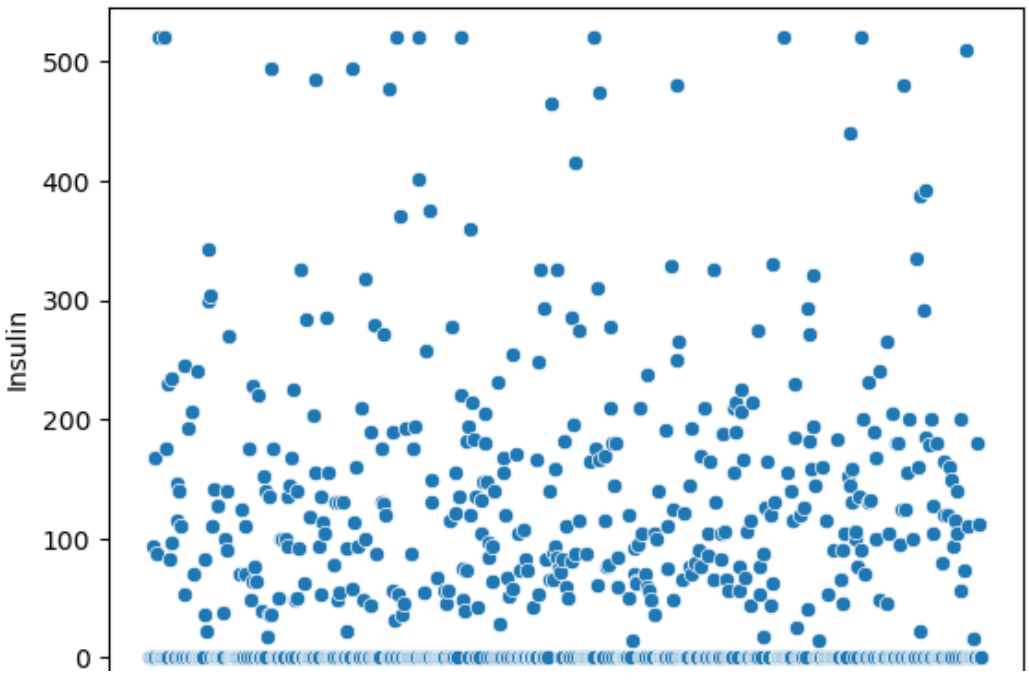
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_22864\2095597971.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  diabetes.Insulin[(diabetes.Insulin>uv)]=uv
```

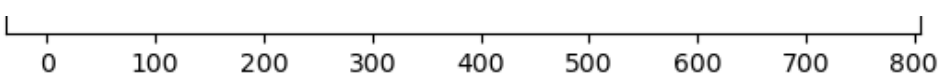
In [131]:

```
sns.scatterplot(diabetes['Insulin'])
```

Out[131]:

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





In [132]:

```
diabetes.describe()
```

Out[132]:

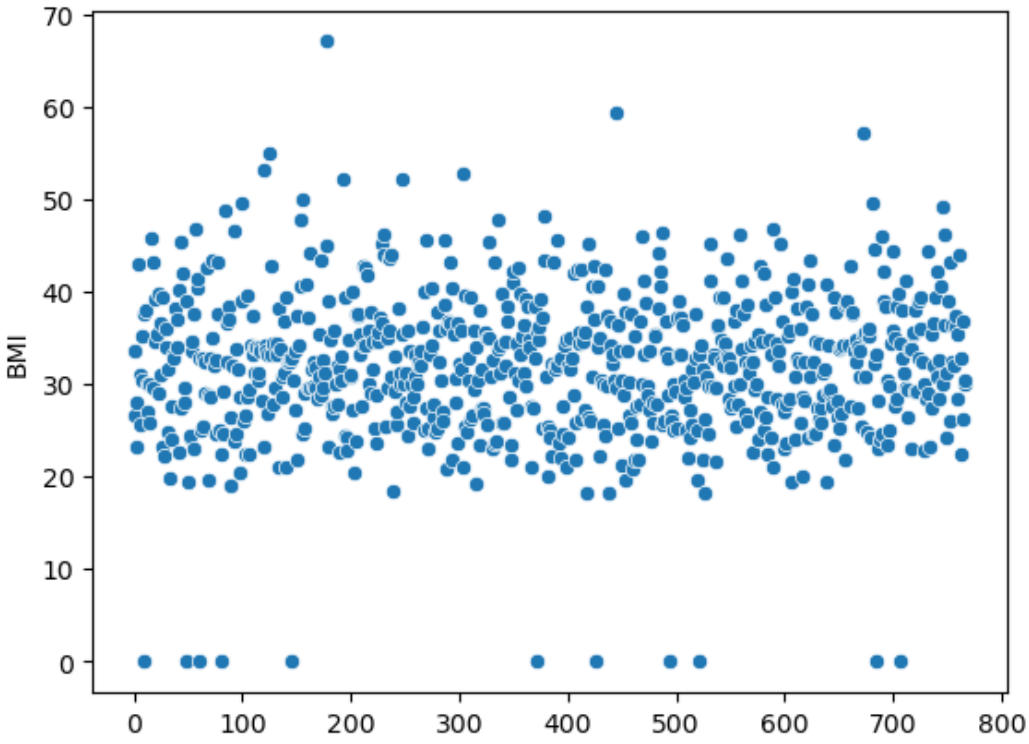
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	78.604427	31.992578	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.952218	109.425722	7.884160	0.331329	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	99.000000	519.900000	67.100000	2.420000	81.000000

In [133]:

```
sns.scatterplot(diabetes.BMI)
```

Out[133]:

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



In [134]:

```
np.percentile(diabetes.BMI,[2])
```

Out[134]:

```
array([19.168])
```

In [135]:

```
lv=np.percentile(diabetes.BMI,[2])[0]
```

In [ ]:

```
```

In [136]:

```
lv1=np.percentile(diabetes.BMI,[99])[0]
```

In [137]:

```
diabetes.BMI[(diabetes.BMI>lv1)]=lv1
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_22864\4237186560.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  diabetes.BMI[(diabetes.BMI>lv1)]=lv1
```

In [138]:

```
diabetes.describe()
```

Out[138]:

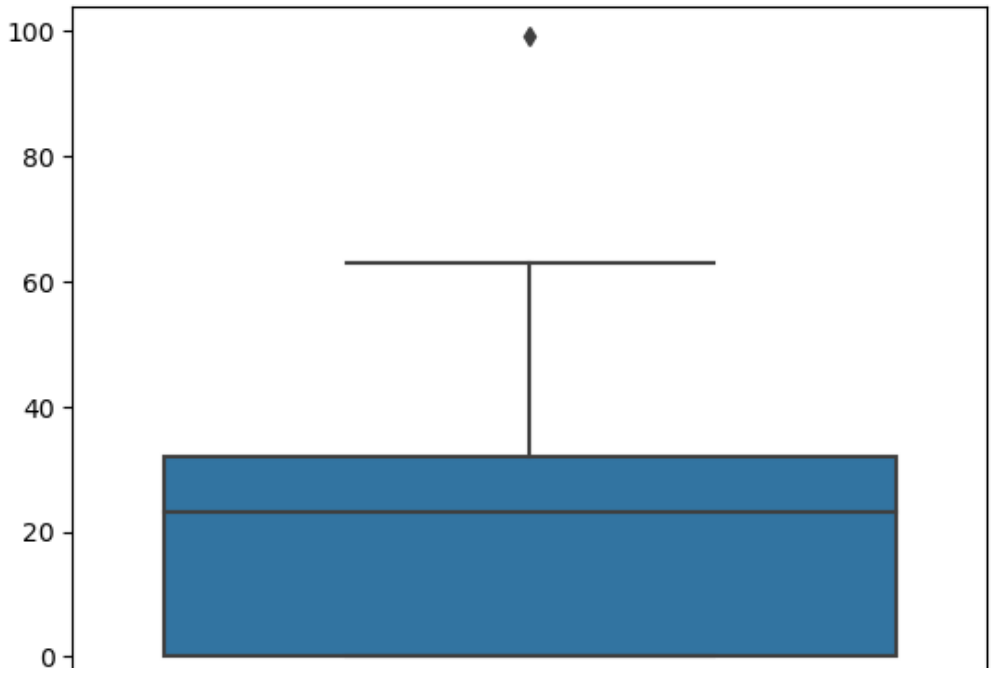
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	78.604427	31.936031	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.952218	109.425722	7.712781	0.331329	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	99.000000	519.900000	50.759000	2.420000	81.000000

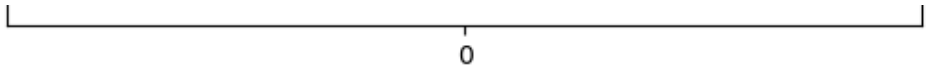
In [139]:

```
sns.boxplot(diabetes['SkinThickness'])
```

Out[139]:

```
<Axes: >
```





In [140]:

```
uv11=np.percentile(diabetes.SkinThickness,[99])[0]
uv11
```

Out[140]:

```
51.33000000000004
```

In [141]:

```
diabetes.SkinThickness[(diabetes.SkinThickness>uv11)]=uv11
```

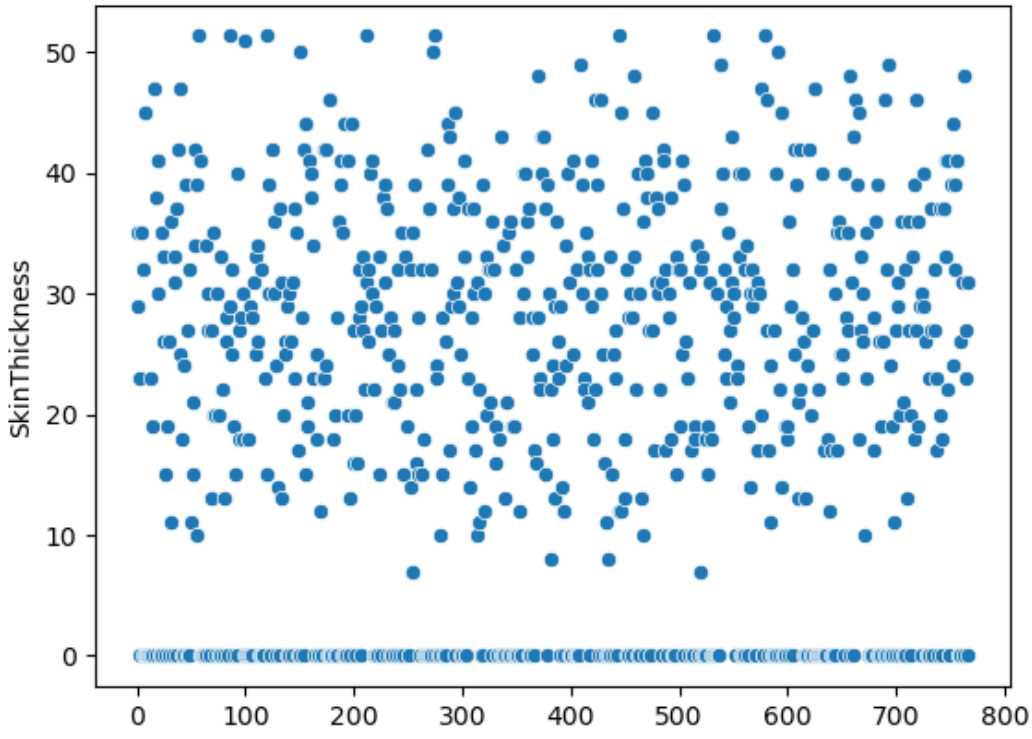
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_22864\236949777.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  diabetes.SkinThickness[(diabetes.SkinThickness>uv11)]=uv11
```

In [142]:

```
sns.scatterplot(diabetes['SkinThickness'])
```

Out[142]:

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



In [143]:

```
diabetes.describe()
```

Out[143]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.433125	78.604427	31.936031	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.646206	109.425722	7.712781	0.331329	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000

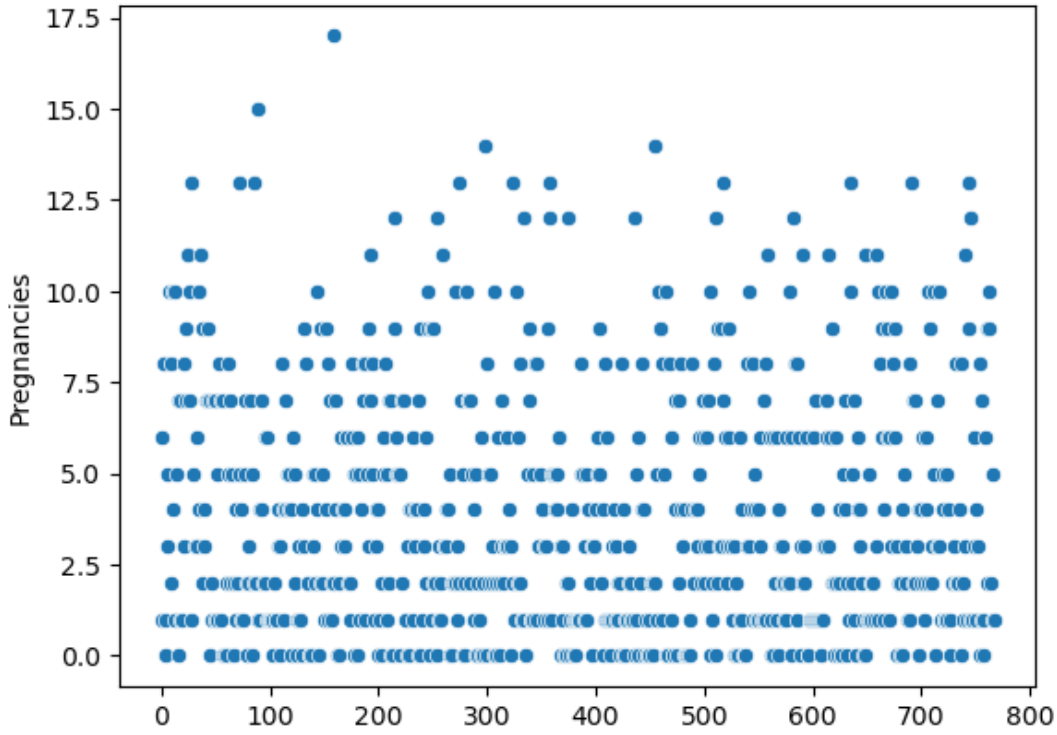
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	51.330000	519.900000	50.759000	2.420000	81.000000

In [144]:

```
sns.scatterplot(diabetes.Pregnancies)
```

Out[144]:

<Axes: ylabel='Pregnancies'>

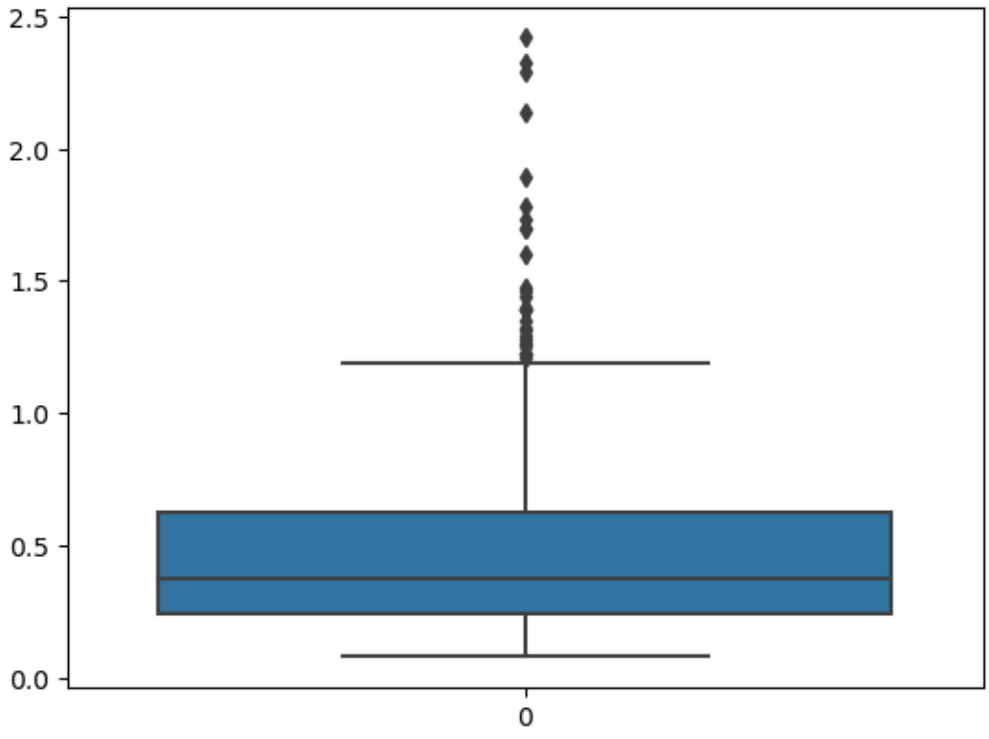


In [145]:

```
sns.boxplot(diabetes.DiabetesPedigreeFunction)
```

Out[145]:

<Axes: >

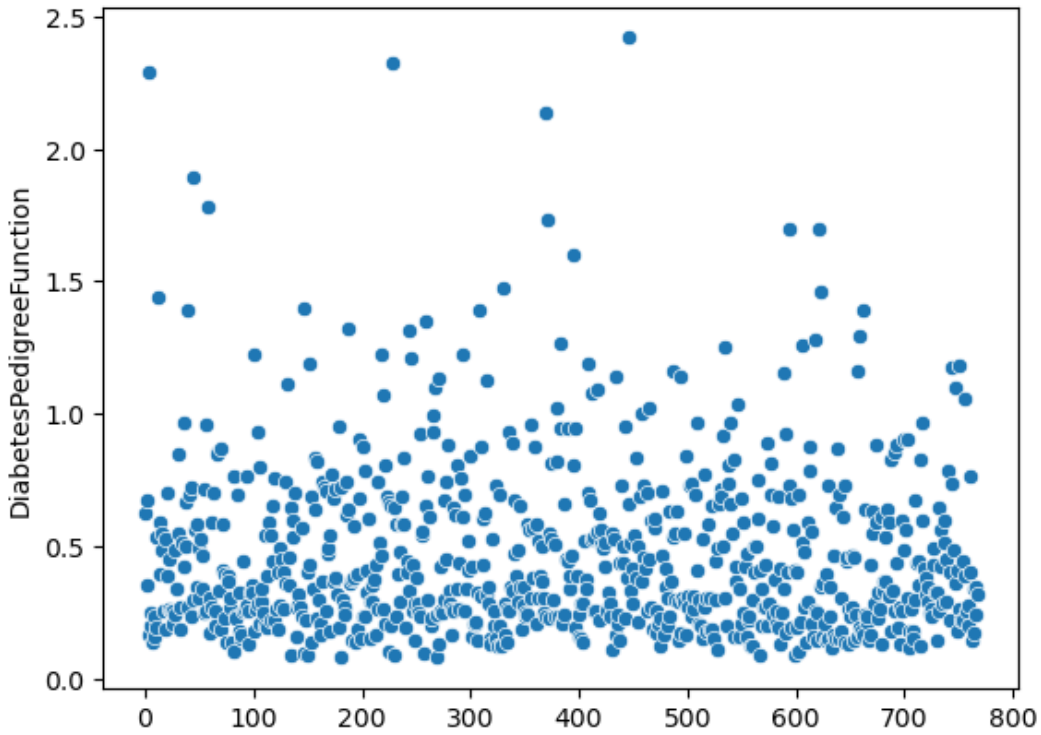


In [146]:

```
sns.scatterplot(diabetes.DiabetesPedigreeFunction)
```

Out[146]:

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



In [147]:

```
uv12=np.percentile(diabetes.DiabetesPedigreeFunction,[99])[0]
uv12
```

Out[147]:

```
1.6983300000000001
```

In [148]:

```
diabetes[(diabetes.DiabetesPedigreeFunction>uv12)]
```

Out[148]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
4	0	137	40	35.00	168.0	43.100	2.288	33	1
45	0	180	66	39.00	0.0	42.000	1.893	25	1
58	0	146	82	0.00	0.0	40.500	1.781	44	0
228	4	197	70	39.00	519.9	36.700	2.329	31	0
370	3	173	82	48.00	465.0	38.400	2.137	25	1
371	0	118	64	23.00	89.0	0.000	1.731	21	0
445	0	180	78	51.33	14.0	50.759	2.420	25	1
593	2	82	52	22.00	115.0	28.500	1.699	25	0

In [149]:

```
diabetes.DiabetesPedigreeFunction[(diabetes.DiabetesPedigreeFunction>uv12)]=uv12
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_22864\3502700536.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  diabetes.DiabetesPedigreeFunction[(diabetes.DiabetesPedigreeFunction>uv12)]=uv12
```

In [150]:

```
diabetes.describe()
```

Out[150]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.433125	78.604427	31.936031	0.468372	33.240885
std	3.369578	31.972618	19.355807	15.646206	109.425722	7.712781	0.314957	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	51.330000	519.900000	50.759000	1.698330	81.000000

In [151]:

```
diabetes.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    float64
 4   Insulin                   768 non-null    float64
 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(4), int64(5)
memory usage: 54.1 KB
```

In [152]:

```
diabetes.columns.tolist()
```

Out[152]:

```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age',
 'Outcome']
```

In [153]:

```
sns.pairplot(diabetes)
```

Out[153]:

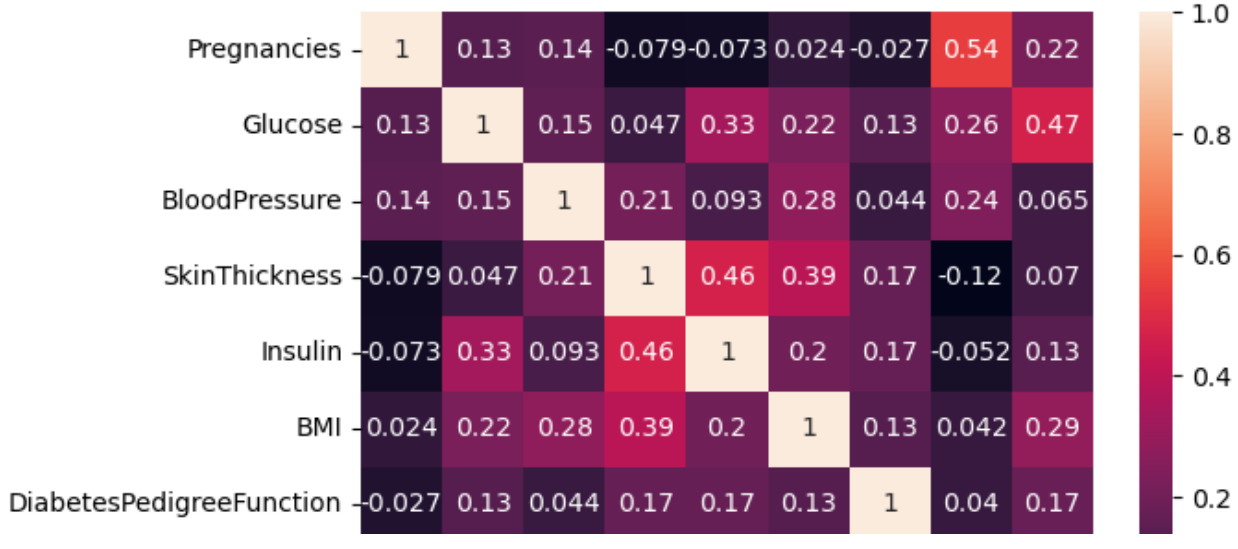
```
<seaborn.axisgrid.PairGrid at 0x19ddb92f8b0>
```

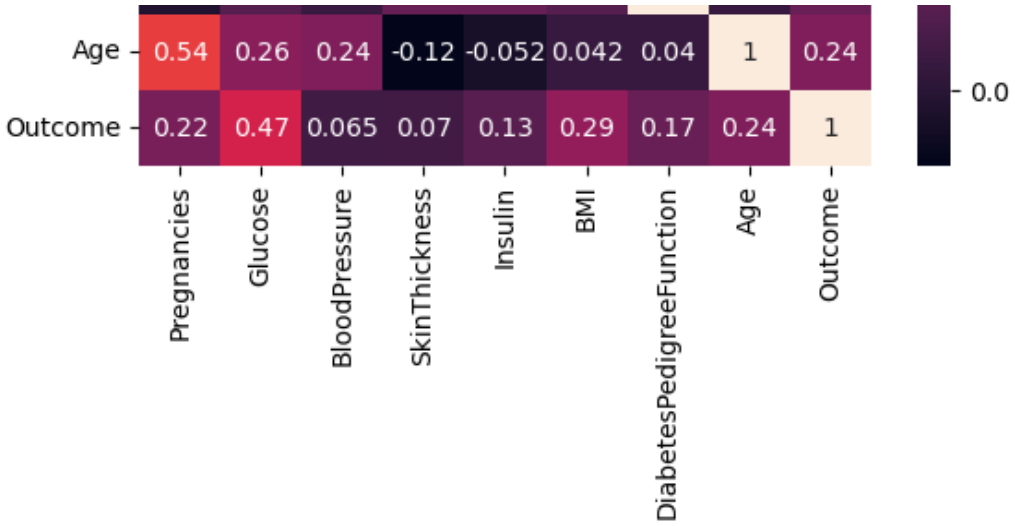
In [154]:

```
sns.heatmap(diabetes.corr(numeric_only=True),annot=True)
```

Out[154]:

<Axes: >





In [155]:

```
diabetes.describe()
```

Out[155]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.433125	78.604427	31.936031	0.468372	33.240885
std	3.369578	31.972618	19.355807	15.646206	109.425722	7.712781	0.314957	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	51.330000	519.900000	50.759000	1.698330	81.000000

In [156]:

```
import statsmodels.api as sm
import pandas as pd

# Assuming you have a DataFrame 'df' with your data
# X should be your feature variables, and y should be your binary target variable

X=sm.add_constant(diabetes[['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age']])
y=diabetes["Outcome"]

# Add a constant term to the features for the intercept
X = sm.add_constant(X)

# Create a logistic regression model
logit_model = sm.Logit(y, X)

# Fit the model to the data
result = logit_model.fit()

# Display the model summary
print(result.summary())
```

```
Optimization terminated successfully.
```

```
         Current function value: 0.471038
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  768
Model:                          Logit   Df Residuals:                      759
Method:                           MLE   Df Model:                            8
Date:                Wed, 27 Dec 2023   Pseudo R-squ.:                  0.2717
Time:                        22:23:04   Log-Likelihood:                -361.76
converged:                       True   LL-Null:                       -496.74
Covariance Type:            nonrobust   LLR p-value:                 9.983e-54
==============================================================================
                                 coef    std err          z      P>|z|      [0.025      0.
975]
------------------------------------------------------------------------------
-----
const                      -8.4831      0.725    -11.696      0.000      -9.905      -7
.062
Pregnancies                 0.1231      0.032      3.835      0.000       0.060       0
.186
Glucose                     0.0350      0.004      9.438      0.000       0.028       0
.042
BloodPressure              -0.0130      0.005     -2.494      0.013      -0.023      -0
.003
SkinThickness               0.0003      0.007      0.049      0.961      -0.014       0
.014
Insulin                    -0.0011      0.001     -1.179      0.238      -0.003       0
.001
BMI                         0.0916      0.016      5.910      0.000       0.061       0
.122
DiabetesPedigreeFunction    1.0087      0.303      3.331      0.001       0.415       1
.602
Age                         0.0145      0.009      1.548      0.122      -0.004       0
.033
==============================================================================
===
```

In [161]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np

# Assuming you have a DataFrame 'df' with your data
# X should be your feature variables, and y should be your target variable (binary)

# Example:
X = diabetes[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI
', 'DiabetesPedigreeFunction', 'Age']]
y = diabetes["Outcome"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a logistic regression model
classifier =  LogisticRegression(random_state=0, max_iter=1000)

# Fit the model to the training data
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Display the number of predictions and actual values
print(np.column_stack((y_pred, y_test)))
```

```
[[1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
```

```
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 1]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[0 1]
[0 1]
[0 1]
[0 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 1]
[0 1]
[1 1]
[0 0]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[0 0]
```

```
[1 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 0]
[0 1]
[1 0]
[1 1]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[1 0]
[0 1]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[1 1]
[0 1]
[1 1]
[1 1]
[0 0]
[1 0]
[0 0]
[1 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[0 1]
[0 0]
[0 1]
[1 1]
[0 0]
[0 0]
[1 0]
[0 0]
[0 0]
[0 1]
[0 0]
[0 0]
[0 0]
[0 0]
```

```
 [0 1]
 [0 0]
 [1 1]
 [0 0]
 [0 0]]
```

In [163]:

```
# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

# Calculate and print accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {accuracy}")
```

```
Confusion Matrix:
[[98  9]
 [18 29]]
Accuracy Score: 0.8246753246753247
```

In [164]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score

# Assuming you have a DataFrame 'diabetes' with your data
# Replace 'your_file.csv' with the actual path to your CSV file if you are reading from a
file

# Read the data or use your existing DataFrame
# df = pd.read_csv('your_file.csv')

# Assuming you already have X and y defined
X = diabetes[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI
', 'DiabetesPedigreeFunction', 'Age']]
y = diabetes["Outcome"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Use StandardScaler to scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create a logistic regression model
classifier = LogisticRegression(random_state=0, solver='lbfgs')

# Fit the model to the scaled training data
classifier.fit(X_train_scaled, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test_scaled)

# Display the number of predictions and actual values
print("Number of Predictions and Actual Values:")
print(pd.DataFrame({'Predicted': y_pred, 'Actual': y_test}).reset_index(drop=True))

# Making the Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Calculate and print accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy Score: {accuracy}")
```

```
Number of Predictions and Actual Values:
     Predicted  Actual
0            1       1
1            0       0
2            0       0
3            1       1
4            0       0
..         ...     ...
149          0       1
150          0       0
151          1       1
152          0       0
153          0       0

[154 rows x 2 columns]

Confusion Matrix:
[[98  9]
 [18 29]]

Accuracy Score: 0.8246753246753247
```

In [165]:

```
import pandas as pd
from sklearn.model_selection import cross_val_score, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import make_scorer, accuracy_score

# Assuming you have a DataFrame 'diabetes' with your data
# Replace 'your_file.csv' with the actual path to your CSV file if you are reading from a
file

# Read the data or use your existing DataFrame
# df = pd.read_csv('your_file.csv')

# Assuming you already have X and y defined
X = diabetes[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI
', 'DiabetesPedigreeFunction', 'Age']]
y = diabetes["Outcome"]

# Use StandardScaler to scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create a logistic regression model
classifier = LogisticRegression(random_state=0, solver='lbfgs')

# Define K-fold cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=0)

# Perform K-fold cross-validation and calculate accuracy
cv_accuracy = cross_val_score(classifier, X_scaled, y, cv=kfold, scoring=make_scorer(acc
uracy_score))

# Display cross-validation results
print("Cross-Validation Accuracy for each fold:")
print(cv_accuracy)
print(f"Mean Accuracy: {cv_accuracy.mean()}")

# Note: You can also access other metrics by changing the 'scoring' parameter in cross_va
l_score
# For example, scoring='precision', scoring='recall', etc.
```

```
Cross-Validation Accuracy for each fold:
[0.82467532 0.77922078 0.74675325 0.75163399 0.77777778]
Mean Accuracy: 0.7760122230710466
```

In [108]:

```
pip install tensorflow
```

```
Collecting tensorflowNote: you may need to restart the kernel to use updated packages.

  Downloading tensorflow-2.15.0-cp310-cp310-win_amd64.whl (2.1 kB)
Collecting tensorflow-intel==2.15.0
  Downloading tensorflow_intel-2.15.0-cp310-cp310-win_amd64.whl (300.9 MB)
     ---------------------------------------- 300.9/300.9 MB 1.5 MB/s eta 0:00:00
Collecting opt-einsum>=2.3.2
  Downloading opt_einsum-3.3.0-py3-none-any.whl (65 kB)
     ---------------------------------------- 65.5/65.5 kB 1.8 MB/s eta 0:00:00
Collecting flatbuffers>=23.5.26
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
Collecting keras<2.16,>=2.15.0
  Downloading keras-2.15.0-py3-none-any.whl (1.7 MB)
     ---------------------------------------- 1.7/1.7 MB 725.6 kB/s eta 0:00:00
Collecting tensorflow-estimator<2.16,>=2.15.0
  Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
     ---------------------------------------- 442.0/442.0 kB 3.5 MB/s eta 0:00:00
Requirement already satisfied: packaging in c:\users\asus\anaconda3\lib\site-packages (fr
om tensorflow-intel==2.15.0->tensorflow) (22.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\asus\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (4.4.0)
Collecting tensorboard<2.16,>=2.15
  Downloading tensorboard-2.15.1-py3-none-any.whl (5.5 MB)
     ---------------------------------------- 5.5/5.5 MB 1.8 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\asus\anaconda3\lib\site-p
ackages (from tensorflow-intel==2.15.0->tensorflow) (1.23.5)
Collecting ml-dtypes~=0.2.0
  Downloading ml_dtypes-0.2.0-cp310-cp310-win_amd64.whl (938 kB)
     ---------------------------------------- 938.6/938.6 kB 550.0 kB/s eta 0:00:00
Collecting absl-py>=1.0.0
  Downloading absl_py-2.0.0-py3-none-any.whl (130 kB)
     ---------------------------------------- 130.2/130.2 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\lib\site-packages (f
rom tensorflow-intel==2.15.0->tensorflow) (65.6.3)
Requirement already satisfied: six>=1.12.0 in c:\users\asus\anaconda3\lib\site-packages (
from tensorflow-intel==2.15.0->tensorflow) (1.16.0)
Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.60.0-cp310-cp310-win_amd64.whl (3.7 MB)
     ---------------------------------------- 3.7/3.7 MB 2.1 MB/s eta 0:00:00
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Requirement already satisfied: h5py>=2.9.0 in c:\users\asus\anaconda3\lib\site-packages (
from tensorflow-intel==2.15.0->tensorflow) (3.7.0)
Collecting libclang>=13.0.0
  Downloading libclang-16.0.6-py2.py3-none-win_amd64.whl (24.4 MB)
     ---------------------------------------- 24.4/24.4 MB 1.1 MB/s eta 0:00:00
Collecting google-pasta>=0.1.1
  Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
     ---------------------------------------- 57.5/57.5 kB 3.0 MB/s eta 0:00:00
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20
.3
  Downloading protobuf-4.25.1-cp310-abi3-win_amd64.whl (413 kB)
     ---------------------------------------- 413.4/413.4 kB 3.7 MB/s eta 0:00:00
Collecting astunparse>=1.6.0
  Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\users\asus\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.15.0->tensorflow) (1.14.1)
Collecting termcolor>=1.1.0
  Downloading termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.31.0-cp310-cp310-win_amd64.whl (1.5 MB)
     ---------------------------------------- 1.5/1.5 MB 926.6 kB/s eta 0:00:00
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\asus\anaconda3\lib\site-pac
kages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow) (0.38.4)
Collecting google-auth-oauthlib<2,>=0.5
  Downloading google_auth_oauthlib-1.2.0-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\asus\anaconda3\lib\site-pa
ckages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.28.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\asus\anaconda3\lib\site-packag
es (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.4.1)
Collecting tensorboard-data-server<0.8.0,>=0.7.0
```

```
  Downloading tensorboard_data_server-0.7.2-py3-none-any.whl (2.4 kB)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\asus\anaconda3\lib\site-packag
es (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (2.2.2)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20
.3
  Downloading protobuf-4.23.4-cp310-abi3-win_amd64.whl (422 kB)
     ---------------------------------------- 422.5/422.5 kB 3.7 MB/s eta 0:00:00
Collecting google-auth<3,>=1.6.3
  Downloading google_auth-2.25.2-py2.py3-none-any.whl (184 kB)
     ---------------------------------------- 184.2/184.2 kB 3.7 MB/s eta 0:00:00
Collecting rsa<5,>=3.1.4
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\asus\anaconda3\lib\site-
packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->
tensorflow) (0.2.8)
Collecting cachetools<6.0,>=2.0.0
  Downloading cachetools-5.3.2-py3-none-any.whl (9.3 kB)
Collecting requests-oauthlib>=0.7.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\asus\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tenso
rflow) (2023.5.7)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\asus\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0-
>tensorflow) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\asus\anaconda3\lib\site-
packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->te
nsorflow) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\anaconda3\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(3.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\asus\anaconda3\lib\site-pack
ages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow
) (2.1.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\asus\anaconda3\lib\site-p
ackages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tens
orflow-intel==2.15.0->tensorflow) (0.4.8)
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
     ---------------------------------------- 151.7/151.7 kB 4.6 MB/s eta 0:00:00
Installing collected packages: libclang, flatbuffers, termcolor, tensorflow-io-gcs-filesy
stem, tensorflow-estimator, tensorboard-data-server, rsa, protobuf, opt-einsum, oauthlib,
ml-dtypes, keras, grpcio, google-pasta, gast, cachetools, astunparse, absl-py, requests-o
authlib, google-auth, google-auth-oauthlib, tensorboard, tensorflow-intel, tensorflow
Successfully installed absl-py-2.0.0 astunparse-1.6.3 cachetools-5.3.2 flatbuffers-23.5.2
6 gast-0.5.4 google-auth-2.25.2 google-auth-oauthlib-1.2.0 google-pasta-0.2.0 grpcio-1.60
.0 keras-2.15.0 libclang-16.0.6 ml-dtypes-0.2.0 oauthlib-3.2.2 opt-einsum-3.3.0 protobuf-
4.23.4 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.15.1 tensorboard-data-server-0.7.2 t
ensorflow-2.15.0 tensorflow-estimator-2.15.0 tensorflow-intel-2.15.0 tensorflow-io-gcs-fi
lesystem-0.31.0 termcolor-2.4.0
```

In [112]:

```
LogisticRegression(random_state=0, max_iter=1000)
```

Out[112]:

```
▾                  LogisticRegression
LogisticRegression(max_iter=1000, random_state=0)
```

In [166]:

```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
```

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

# Split the dataset into training and testing sets
X = diabetes[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI
', 'DiabetesPedigreeFunction', 'Age']]
y = diabetes["Outcome"]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Define classifiers
classifiers = [
    ('Logistic Regression', LogisticRegression(random_state=0, max_iter=1000)),
    ('Random Forest', RandomForestClassifier(random_state=0)),
    ('Support Vector Machine', SVC(kernel='rbf', random_state=0)),
    ('K-Nearest Neighbors', KNeighborsClassifier(n_neighbors=5)),
    ('Decision Tree', DecisionTreeClassifier(random_state=0)),
    ('XGBoost', XGBClassifier(random_state=0)),
    ('Naive Bayes', GaussianNB()),

]
# Train and evaluate each classifier
for clf_name, clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Evaluate the classifier
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    # Display results
    print(f"Classifier: {clf_name}")
    print(f"Accuracy: {accuracy:.4f}")
    print("Classification Report:\n", report)
    print("="*50)
```

```
Classifier: Logistic Regression
Accuracy: 0.8247
Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.92      0.88       107
           1       0.76      0.62      0.68        47

    accuracy                           0.82       154
   macro avg       0.80      0.77      0.78       154
weighted avg       0.82      0.82      0.82       154

==================================================
Classifier: Random Forest
Accuracy: 0.7792
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.86      0.84       107
           1       0.65      0.60      0.62        47

    accuracy                           0.78       154
   macro avg       0.74      0.73      0.73       154
weighted avg       0.77      0.78      0.78       154

==================================================
Classifier: Support Vector Machine
Accuracy: 0.7922
Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.92      0.86       107
           1       0.73      0.51      0.60        47

    accuracy                           0.79       154
```

```
   macro avg       0.77      0.71      0.73       154
weighted avg       0.78      0.79      0.78       154

==================================================
Classifier: K-Nearest Neighbors
Accuracy: 0.7468
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.81      0.82       107
           1       0.58      0.60      0.59        47

    accuracy                           0.75       154
   macro avg       0.70      0.70      0.70       154
weighted avg       0.75      0.75      0.75       154

==================================================
Classifier: Decision Tree
Accuracy: 0.7597
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.81      0.82       107
           1       0.60      0.64      0.62        47

    accuracy                           0.76       154
   macro avg       0.72      0.73      0.72       154
weighted avg       0.76      0.76      0.76       154

==================================================
Classifier: XGBoost
Accuracy: 0.7792
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.84      0.84       107
           1       0.64      0.64      0.64        47

    accuracy                           0.78       154
   macro avg       0.74      0.74      0.74       154
weighted avg       0.78      0.78      0.78       154

==================================================
Classifier: Naive Bayes
Accuracy: 0.7922
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       107
           1       0.67      0.62      0.64        47

    accuracy                           0.79       154
   macro avg       0.76      0.74      0.75       154
weighted avg       0.79      0.79      0.79       154

==================================================
```

In [ ]: