Group B - ASSIGNMENT NO 10

Title - Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

```
# Name - Vedant Kulkarni
# Roll Number - 51

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, f1_score, recall_score, 
precision_score,accuracy_score
```

```
df=pd.read_csv("C:\\Users\\Asus\\Downloads\\diabetes.csv")
```

```
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


   Pedigree  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.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    Pedigree         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
```

```
#replace zeros
zero_not_accepted=["Glucose","BloodPressure","SkinThickness","BMI","In
sulin"]
for column in zero_not_accepted:
    df[column]=df[column].replace(0,np.NaN)
    mean=int(df[column].mean(skipna=True))
    df[column]=df[column].replace(np.NaN,mean)

df["Glucose"]
```

```
0      148.0
1       85.0
2      183.0
3       89.0
4      137.0
       ...
763    101.0
764    122.0
765    121.0
766    126.0
```

```
767      93.0
Name: Glucose, Length: 768, dtype: float64
```

```
#split dataset
X=df.iloc[:,0:8]
y=df.iloc[:,8]
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0,test
_size=0.2)
```

```
#feature Scaling
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)
```

```
X_test=sc_X.transform(X_test)
```

```
knn=KNeighborsClassifier(n_neighbors=11)
```

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=11)
```

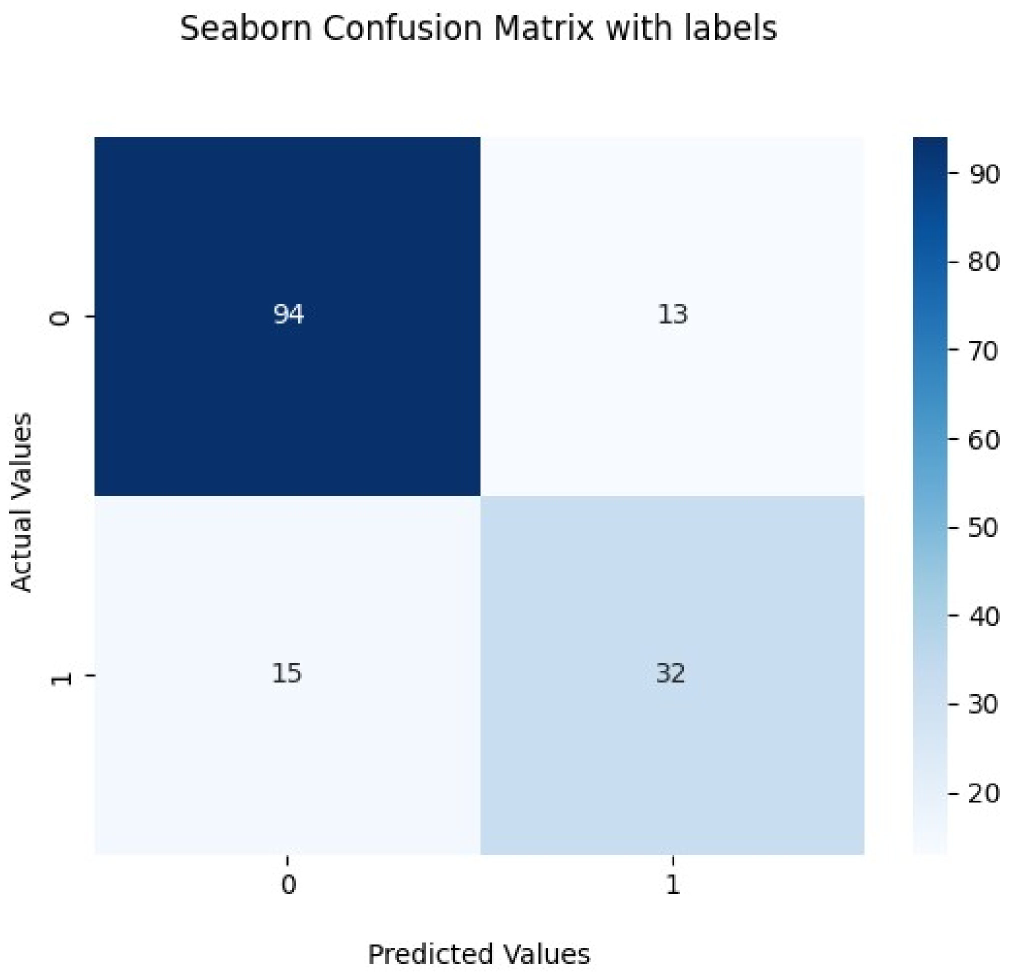
```
y_pred=knn.predict(X_test)
```

```
#Evaluate The Model
cf_matrix=confusion_matrix(y_test,y_pred)
```

```
ax = sns.heatmap(cf_matrix, annot=True, cmap='Blues')
```

```
ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');


## Display the visualization of the Confusion Matrix.
plt.show()
```



```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred ).ravel()

tn, fp, fn, tp

(94, 13, 15, 32)

#The accuracy rate is equal to (tn+tp)/(tn+tp+fn+fp)
accuracy_score(y_test,y_pred)

0.8181818181818182

#The precision is the ratio of tp/(tp + fp)
precision_score(y_test,y_pred)

0.7111111111111111

##The recall is the ratio of tp/(tp + fn)
recall_score(y_test,y_pred)

0.6808510638297872
```

```
#error rate=1-accuracy which is lies bertween 0 and 1
error_rate=1-accuracy_score(y_test,y_pred)

error_rate
```

```
0.18181818181818177
```