Name :: Gayatri Manikchand Deshmukh
Class :: BE A
Roll No.:: 38

Problem Statement

Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.
Dataset link : https://www.kaggle.com/datasets/abdallamahgoub/diabetes

In [10]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:

```
df = pd.read_csv("diabetes.csv")
```

In [5]:

```
df.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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 [6]:

```
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   Pedigree       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 [7]:

```
df.isnull().sum()
```

Out[7]:

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

In [9]:

```
df.columns
```

Out[9]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'Pedigree', 'Age', 'Outcome'],
      dtype='object')
```

In [11]:

```
#replace zeros
zero_not_accepted=["Glucose","BloodPressure","SkinThickness","BMI","Insulin"]
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)
```

In [12]:

```
df['Glucose']
```

Out[12]:

```
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
```

In [16]:

```
#split dataset
from sklearn.model_selection import train_test_split
x=df.iloc[:,0:8]
y=df.iloc[:,8]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0,test_size=0.2)
```

In [18]:

```
from sklearn.preprocessing import StandardScaler
#feature Scaling
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
xtest=sc.transform(xtest)
```

In [19]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=11)
```

In [20]:

```
knn.fit(xtrain,ytrain)
```

Out[20]:

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

In [21]:

```
ypred=knn.predict(xtest)
```

In [22]:

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
```

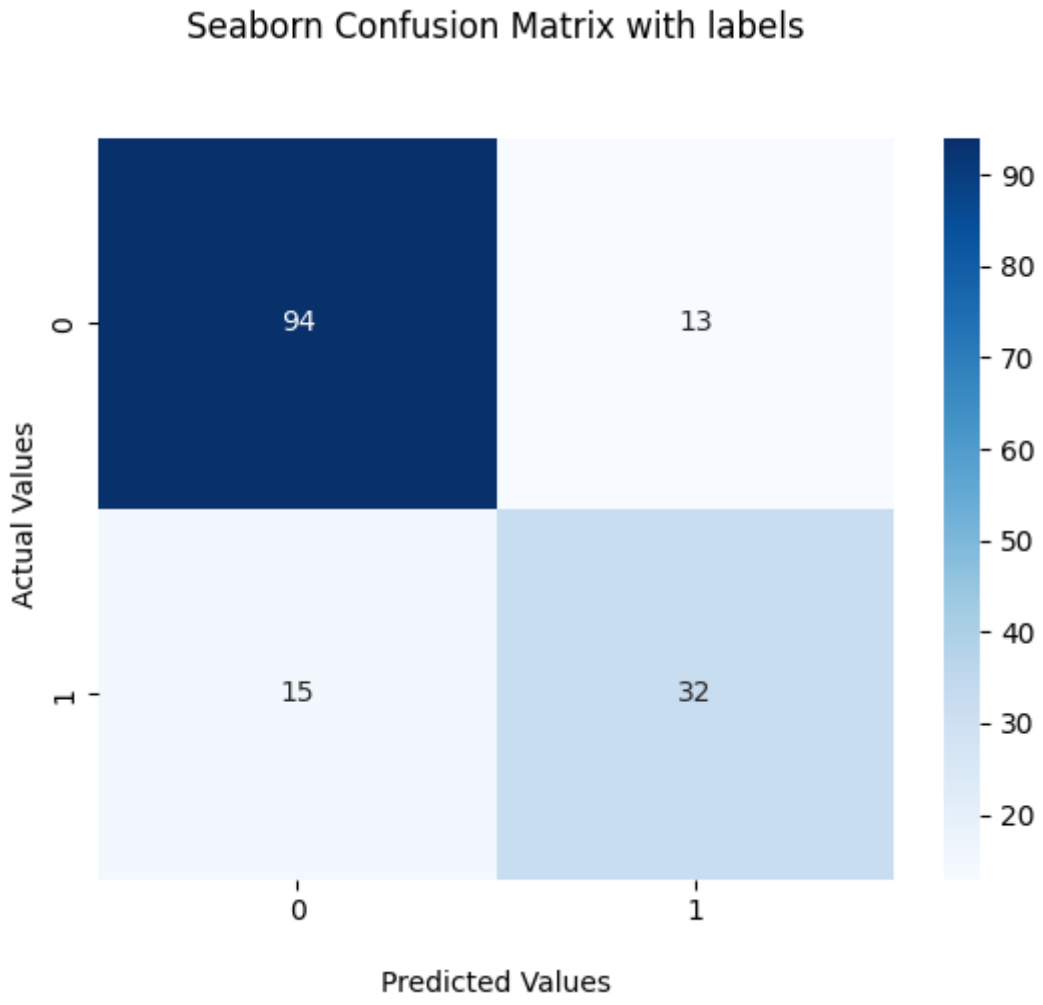
In [24]:

```
cm = confusion_matrix(ytest, ypred)
print("Confusion Matrix \n",cm)
```

```
Confusion Matrix
 [[94 13]
 [15 32]]
```

In [25]:

```
ax = sns.heatmap(cm, annot=True, cmap='Blues')
ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
```



In [26]:

```
tn, fp, fn, tp = confusion_matrix(ytest, ypred ).ravel()
```

In [27]:

```
print("Accuracy Score :: ",accuracy_score(ytest, ypred))
```

```
Accuracy Score ::  0.8181818181818182
```

In [28]:

```
print("Precision Score :: ",precision_score(ytest, ypred))
```

```
Precision Score ::  0.7111111111111111
```

In [29]:

```
print("Recall Score :: ",recall_score(ytest, ypred))
```

```
Recall Score ::  0.6808510638297872
```

In [30]:

```
print("Error rate :: ",1-accuracy_score(ytest, ypred))
```

```
Error rate ::  0.18181818181818177
```

In [ ]:

```

```