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 [1]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("diabetes.csv")
```

In [2]:
```
df.head()
```

Out[2]:

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

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

decide the dependent and independent variable

In [4]:
```
X=df.drop(['Outcome'],axis=1)
Y=df['Outcome']
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X=sc.fit_transform(X)
X
```

Out[10]:
```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=2)
```

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
y_pred=knn.predict(X_test)
y_pred
```

Out[19]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

In [20]:
```
from sklearn import metrics
print("Confusion Matrix \n",metrics.confusion_matrix(Y_test,y_pred))
```

```
Confusion Matrix
 [[136  19]
 [ 44  32]]
```

In [21]:
```
print("Accuracy Score \n",metrics.accuracy_score(Y_test,y_pred))
```

```
Accuracy Score
 0.7272727272727273
```

In [22]:
```
print("Precision Score \n",metrics.precision_score(Y_test,y_pred))
```

```
Precision Score
 0.6274509803921569
```

In [23]:
```
print("Recall Score \n",metrics.recall_score(Y_test,y_pred))
```

```
Recall Score
 0.42105263157894735
```

In [26]:
```
print("error rate \n",1-metrics.accuracy_score(Y_test,y_pred))
```

```
error rate
 0.2727272727272727
```

In [24]:
```
print("Classification Report \n",metrics.classification_report(Y_test,y_pred))
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.88      0.81       155
           1       0.63      0.42      0.50        76

    accuracy                           0.73       231
   macro avg       0.69      0.65      0.66       231
weighted avg       0.71      0.73      0.71       231
```

In [ ]:
```

```