In [1]:
```
#without using library

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report

data = pd.read_csv(r"C:\Program Files\Python311\Scripts\iris.csv",header='infer').values
x = data[:,1:-1]
y = data[:,-1]
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,shuffle=True)
print("Shapes are: ",x_train.shape,x_test.shape,y_train.shape,y_test.shape)
nclasses = np.unique(y_train).shape[0]
print(nclasses)

dist = np.zeros(shape = x_train.shape[0])
pred = np.zeros(shape = x_test.shape[0])
classvotes = np.zeros(shape = nclasses)

k = int(input("Enter the number of nearest neighbours to be used, i.e. k: "))
for i in range(x_test.shape[0]):
    dist = np.sqrt(np.sum((x_train-x_test[i])**2,axis = 1))
    kminind = np.argpartition(dist,k)[0:k]
    invdist = 1/(dist+10e-20)
    denom = sum(invdist[kminind])
    for j in range(k):
        classvotes[int(y_train[kminind[j]])] += invdist[kminind[j]]
    classvotes /= denom
    pred[i] = np.argmax(classvotes)

print(pred)
def calc_acc(y_pred,y_true):
    return np.sum((y_pred).astype(int) == (y_true).astype(int))/y_pred.shape[0]
def calc_prec(y_pred,y_true):
    classes = np.unique(y_true)
    nclasses = classes.shape[0]
    nrows = y_true.shape[0]
    classprop = np.zeros(shape=nclasses)
    for i in range(nclasses):
        classprop[i] = np.sum(y_true == classes[i])/nrows
    precclasswise = np.zeros(shape=nclasses)
    prec = 0
    for i in range(nclasses):
        predindices = np.where((((y_pred).astype(int)) == (classes[i].astype(int))) == True)
        trueindices = np.where((((y_true).astype(int)) == (classes[i].astype(int))) == True)
        precclasswise[i] = ((len(predindices[0]))-(len(set(predindices[0])-set(trueindices[0]))))/(len(predindices[0]))
        prec += precclasswise[i]*classprop[i]
    return prec

def calc_recall(y_pred,y_true):
    classes = np.unique(y_true)
    nclasses = classes.shape[0]
    nrows = y_true.shape[0]
    classprop = np.zeros(shape=nclasses)
    for i in range(nclasses):
        classprop[i] = np.sum(y_true == classes[i])/nrows
    recallclasswise=np.zeros(shape==nclasses)
    recall=0
    for i in range(nclasses):
        predindices = np.where((((y_pred).astype(int)) == (classes[i].astype(int))) == True)
        trueindices = np.where((((y_true).astype(int)) == (classes[i].astype(int))) == True)
        recallclasswise[i] = ((len(predindices[0]))-(len(set(trueindices[0])-set(predindices[0]))))/(len(predindices[0]))
        recall += recallclasswise[i]*classprop[i]
    return recall

accuracy=calc_acc(pred,y_test)
prec=calc_prec(pred,y_test)
recall=calc_prec(pred,y_test)
f1score=(2*prec*recall)/(prec+recall)

print("Accuracy: ",accuracy)
print("Precision: ",prec)
print("Recall: ",recall)
print("F1-score: ",f1score)

print("Classification Report :")
print(classification_report(y_test,pred))
```

```
Shapes are:  (120, 4) (30, 4) (120,) (30,)
3
Enter the number of nearest neighbours to be used, i.e. k: 3
[2. 1. 1. 2. 0. 0. 0. 2. 0. 1. 0. 2. 2. 2. 0. 2. 1. 1. 1. 0. 2. 2. 1. 1.
 1. 0. 1. 0. 1. 2.]
Accuracy:  0.9666666666666667
Precision:  0.9696969696969697
Recall:  0.9696969696969697
F1-score:  0.9696969696969697
Classification Report :
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         9
         1.0       0.91      1.00      0.95        10
         2.0       1.00      0.91      0.95        11

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

In [2]:
```
#with library
#using sklearn

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report
from sklearn.neighbors import KNeighborsClassifier

data=pd.read_csv(r"C:\Program Files\Python311\Scripts\iris.csv",header='infer').values
x=data[:,1:-1]
y=data[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,shuffle=True)
k=int(input("Enter number of nearest neighbors: "))

model=KNeighborsClassifier(n_neighbors=k,weights='distance')
model.fit(x_train,y_train)
pred=model.predict(x_test)
accuracy=accuracy_score(y_test,pred)

print('Accuracy: ',accuracy)
print(classification_report(y_test,pred))
```

```
Enter number of nearest neighbors: 3
Accuracy:  0.9333333333333333
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        11
         1.0       1.00      0.80      0.89        10
         2.0       0.82      1.00      0.90         9

    accuracy                           0.93        30
   macro avg       0.94      0.93      0.93        30
weighted avg       0.95      0.93      0.93        30
```

In [ ]:

In [ ]:

In [ ]: