In [1]:

```
import numpy as np
import pandas as pd
from scipy.stats import mode
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
from numpy.random import randint
```

In [2]:

```
iris = load_iris()
```

In [3]:

```
X = iris.data
Y = iris.target
```

Splitting Train-Test dataset

In [4]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test  = train_test_split(X, Y, test_size=0.2, random_state=1
0)
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(120, 4) (30, 4) (120,) (30,)
```

Creating Function

In [5]:

```
#Euclidean Distance
def eucledian(p1,p2):
    dist = np.sqrt(np.sum((p1-p2)**2))
    return dist

#Function to calculate KNN
def knn_predict(x_train, y , x_input, k):
    op_labels = []

    #Loop through the Datapoints to be classified
    for item in x_input:
        #Array to store distances
        point_dist = [eucledian(np.array(x_train[j,:]) , item)  for j in range(len(x_tra
in))]
        point_dist = np.array(point_dist)

        #Sorting the array while preserving the index
        #Keeping the first K datapoints in variable 'dist'
        dist = np.argsort(point_dist)[:k]

        #Labels of the K datapoints from above
        labels = y[dist]

        #Majority voting
        lab = mode(labels)[0]
        op_labels.append(lab)

    return op_labels
```

In [6]:

```
#Applying our function
Y_pred = knn_predict(X_train, Y_train, X_test , 5)
```

sklearn

In [7]:

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=5)
```

In [8]:

```
model.fit(X_train, Y_train)
```

Out[8]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [9]:

```
Y_pred_sk = model.predict(X_test)
```

Comparison

In [10]:

```
print(f'Accuracy using self-made function : {accuracy_score(Y_test, Y_pred)}')
print(f'Accuracy using sklearn : {accuracy_score(Y_test, Y_pred_sk)}')
```

```
Accuracy using self-made function : 0.9666666666666667
Accuracy using sklearn : 0.9666666666666667
```

In [10]:

```

```