**Name: YATHIN K V**

**USN: 1PE17IS105**

**Class: ISE 7B**

**Machine Learning Laboratory**

Program 9

Write a program to implement ***k*-Nearest Neighbour algorithm** to classify the iris data

set. Print both correct and wrong predictions. Java/Python ML library classes can be

used for this problem.

K-Nearest-Neighbour Algorithm :

1. Load the data

2. Initialize the value of k

3. For getting the predicted class, iterate from 1 to total number of training data points

1. Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it’s the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.

2. Sort the calculated distances in ascending order based on distance values

3. Get top k rows from the sorted array

4. Get the most frequent class of these rows i.e Get the labels of the selected K entries

5. Return the predicted class

• If regression, return the mean of the K labels

• If classification, return the mode of the K labels

**PROGRAM:**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn import metrics

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = pd.read\_csv("Data\_8\_9.csv", names = names)

print(dataset.head())

x = dataset.iloc[:, :-1].values

y = dataset.iloc[:,4].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size =20 )

scaler = StandardScaler()

scaler.fit(x\_train)

x\_train = scaler.transform(x\_train)

x\_test = scaler.transform(x\_test)

classifier = KNeighborsClassifier(n\_neighbors = 5)

classifier.fit(x\_train, y\_train)

y\_pred = classifier.predict(x\_test)

for i in range(len(y\_pred)):

print ("Training Example : ")

print(x\_test[i])

print ("Actual Label : ")

print(y\_test[i])

print ("Predicted Label : ")

print (y\_pred[i])

print ("--------------------------------------------")

print ("Confusion Matrix : ")

print(confusion\_matrix(y\_test, y\_pred))

print ("")

print(metrics.accuracy\_score(y\_test, y\_pred)\*100)

print ("Classification Report : ")

print(classification\_report(y\_test, y\_pred))

**OUTPUT:**

