Pgm 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.

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

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

iris\_dataset=load\_iris()

print("\n IRIS FEATURES \ TARGET NAMES: \n ", iris\_dataset.target\_names)

for i in range(len(iris\_dataset.target\_names)):

print("\n[{0}]:[{1}]".format(i,iris\_dataset.target\_names[i]))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(iris\_dataset["data"], iris\_dataset["target"], random\_state=0)

print("\n X TRAIN \n", X\_train)

print("\n X TEST \n", X\_test)

print("\n Y TRAIN \n", y\_train)

print("\n Y TEST \n", y\_test)

kn = KNeighborsClassifier(n\_neighbors=1)

kn.fit(X\_train, y\_train)

x\_new = np.array([[5, 2.9, 1, 0.2]])

print("\n XNEW \n",x\_new)

prediction = kn.predict(x\_new)

print("\n Predicted target value: {}\n".format(prediction))

print("\n Predicted feature name: {}\n".format

(iris\_dataset["target\_names"][prediction]))

i=1

x= X\_test[i]

x\_new = np.array([x])

print("\n XNEW \n",x\_new)

for i in range(len(X\_test)): x = X\_test[i]

x\_new = np.array([x])

prediction = kn.predict(x\_new)

print("\nActual:{0}{1},Predicted:{2}{3}".format(y\_test[i],iris\_dataset["target\_names"][y\_test[i]], prediction,iris\_dataset["target\_names"][ prediction]))

print("\n TEST SCORE[ACCURACY]: {:.2f}\n".format(kn.score(X\_test, y\_test)))

**Output**

IRIS FEATURES \ TARGET NAMES:

['setosa' 'versicolor' 'virginica']

[0]:[setosa]

[1]:[versicolor]

[2]:[virginica]

X TRAIN

[[5.9 3. 4.2 1.5]

[5.8 2.6 4. 1.2]

[6.8 3. 5.5 2.1]

[4.7 3.2 1.3 0.2]

[6.9 3.1 5.1 2.3]

[5. 3.5 1.6 0.6]

[5.4 3.7 1.5 0.2]

[5. 2. 3.5 1. ]

[6.5 3. 5.5 1.8]

[6.7 3.3 5.7 2.5]

[6. 2.2 5. 1.5]

[6.7 2.5 5.8 1.8]

[5.6 2.5 3.9 1.1]

[7.7 3. 6.1 2.3]

[6.3 3.3 4.7 1.6]

[5.5 2.4 3.8 1.1]

[6.3 2.7 4.9 1.8]

[6.3 2.8 5.1 1.5]

[4.9 2.5 4.5 1.7]

[6.3 2.5 5. 1.9]

[7. 3.2 4.7 1.4]

[6.5 3. 5.2 2. ]

[6. 3.4 4.5 1.6]

[4.8 3.1 1.6 0.2]

[5.8 2.7 5.1 1.9]

[5.6 2.7 4.2 1.3]

[5.6 2.9 3.6 1.3]

[5.5 2.5 4. 1.3]

[6.1 3. 4.6 1.4]

[7.2 3.2 6. 1.8]

[5.3 3.7 1.5 0.2]

[4.3 3. 1.1 0.1]

[6.4 2.7 5.3 1.9]

[5.7 3. 4.2 1.2]

[5.4 3.4 1.7 0.2]

[5.7 4.4 1.5 0.4]

[6.9 3.1 4.9 1.5]

[4.6 3.1 1.5 0.2]

[5.9 3. 5.1 1.8]

[5.1 2.5 3. 1.1]

[4.6 3.4 1.4 0.3]

[6.2 2.2 4.5 1.5]

[7.2 3.6 6.1 2.5]

[5.7 2.9 4.2 1.3]

[4.8 3. 1.4 0.1]

[7.1 3. 5.9 2.1]

[6.9 3.2 5.7 2.3]

[6.5 3. 5.8 2.2]

[6.4 2.8 5.6 2.1]

[5.1 3.8 1.6 0.2]

[4.8 3.4 1.6 0.2]

[6.5 3.2 5.1 2. ]

[6.7 3.3 5.7 2.1]

[4.5 2.3 1.3 0.3]

[6.2 3.4 5.4 2.3]

[4.9 3. 1.4 0.2]

[5.7 2.5 5. 2. ]

[6.9 3.1 5.4 2.1]

[4.4 3.2 1.3 0.2]

[5. 3.6 1.4 0.2]

[7.2 3. 5.8 1.6]

[5.1 3.5 1.4 0.3]

[4.4 3. 1.3 0.2]

[5.4 3.9 1.7 0.4]

[5.5 2.3 4. 1.3]

[6.8 3.2 5.9 2.3]

[7.6 3. 6.6 2.1]

[5.1 3.5 1.4 0.2]

[4.9 3.1 1.5 0.2]

[5.2 3.4 1.4 0.2]

[5.7 2.8 4.5 1.3]

[6.6 3. 4.4 1.4]

[5. 3.2 1.2 0.2]

[5.1 3.3 1.7 0.5]

[6.4 2.9 4.3 1.3]

[5.4 3.4 1.5 0.4]

[7.7 2.6 6.9 2.3]

[4.9 2.4 3.3 1. ]

[7.9 3.8 6.4 2. ]

[6.7 3.1 4.4 1.4]

[5.2 4.1 1.5 0.1]

[6. 3. 4.8 1.8]

[5.8 4. 1.2 0.2]

[7.7 2.8 6.7 2. ]

[5.1 3.8 1.5 0.3]

[4.7 3.2 1.6 0.2]

[7.4 2.8 6.1 1.9]

[5. 3.3 1.4 0.2]

[6.3 3.4 5.6 2.4]

[5.7 2.8 4.1 1.3]

[5.8 2.7 3.9 1.2]

[5.7 2.6 3.5 1. ]

[6.4 3.2 5.3 2.3]

[6.7 3. 5.2 2.3]

[6.3 2.5 4.9 1.5]

[6.7 3. 5. 1.7]

[5. 3. 1.6 0.2]

[5.5 2.4 3.7 1. ]

[6.7 3.1 5.6 2.4]

[5.8 2.7 5.1 1.9]

[5.1 3.4 1.5 0.2]

[6.6 2.9 4.6 1.3]

[5.6 3. 4.1 1.3]

[5.9 3.2 4.8 1.8]

[6.3 2.3 4.4 1.3]

[5.5 3.5 1.3 0.2]

[5.1 3.7 1.5 0.4]

[4.9 3.1 1.5 0.1]

[6.3 2.9 5.6 1.8]

[5.8 2.7 4.1 1. ]

[7.7 3.8 6.7 2.2]

[4.6 3.2 1.4 0.2]]

X TEST

[[5.8 2.8 5.1 2.4]

[6. 2.2 4. 1. ]

[5.5 4.2 1.4 0.2]

[7.3 2.9 6.3 1.8]

[5. 3.4 1.5 0.2]

[6.3 3.3 6. 2.5]

[5. 3.5 1.3 0.3]

[6.7 3.1 4.7 1.5]

[6.8 2.8 4.8 1.4]

[6.1 2.8 4. 1.3]

[6.1 2.6 5.6 1.4]

[6.4 3.2 4.5 1.5]

[6.1 2.8 4.7 1.2]

[6.5 2.8 4.6 1.5]

[6.1 2.9 4.7 1.4]

[4.9 3.6 1.4 0.1]

[6. 2.9 4.5 1.5]

[5.5 2.6 4.4 1.2]

[4.8 3. 1.4 0.3]

[5.4 3.9 1.3 0.4]

[5.6 2.8 4.9 2. ]

[5.6 3. 4.5 1.5]

[4.8 3.4 1.9 0.2]

[4.4 2.9 1.4 0.2]

[6.2 2.8 4.8 1.8]

[4.6 3.6 1. 0.2]

[5.1 3.8 1.9 0.4]

[6.2 2.9 4.3 1.3]

[5. 2.3 3.3 1. ]

[5. 3.4 1.6 0.4]

[6.4 3.1 5.5 1.8]

[5.4 3. 4.5 1.5]

[5.2 3.5 1.5 0.2]

[6.1 3. 4.9 1.8]

[6.4 2.8 5.6 2.2]

[5.2 2.7 3.9 1.4]

[5.7 3.8 1.7 0.3]

[6. 2.7 5.1 1.6]]

Y TRAIN

[1 1 2 0 2 0 0 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 0 2 1 1 1 1 2 0 0 2 1 0 0 1

0 2 1 0 1 2 1 0 2 2 2 2 0 0 2 2 0 2 0 2 2 0 0 2 0 0 0 1 2 2 0 0 0 1 1 0 0

1 0 2 1 2 1 0 2 0 2 0 0 2 0 2 1 1 1 2 2 1 1 0 1 2 2 0 1 1 1 1 0 0 0 2 1 2 0]

Y TEST

[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 1]

XNEW

[[5. 2.9 1. 0.2]]

Predicted target value: [0]

Predicted feature name: ['setosa']

XNEW

[[6.1 2.6 5.6 1.4]]

Actual : 1 versicolor, Predicted :[2]['virginica']

TEST SCORE[ACCURACY]: 0.97