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.

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
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]))
print(" \n IRIS DATA:\n", iris_dataset["data"])

x_train, x_test, y_train, y_test = train_test_split (iris_dataset
                      ["data"], iris_dataset['target'],
                                    random_state=0)

classifier= KNeighborsClassifier(n_neighbors=8, p=3, metric='euclidean')
classifier.fit( x_train, y_train)
```

```
y-pred = classifier.predict(x.test)
cm = confusion-matrix(y-test, y-pred)
print(' Confusion matrix is as follows\n', cm)
print('Accuracy metrics')
print(classification-report(y-test, y-pred)
print("correction prediction", accuracy-score(y-test, y-pred))
print("wrong prediction", (1-accuracy-score (y-test ,y-pred))
```

Output

```
IRIS FEATURES \ TARGET NAMES :
['setosa' 'versicolor' 'virginica']
 [0]: [setosa]
 [1]: [versicolor]
 [2]: [virginica]

IRIS DATA:
    [ [5.1   3.5   1.4   0.2]
      [4.9   3.0   1.4   0.2]
      [4.7   3.2   1.3   0.2]
           .
           .
           .

KNeighborsClassifier (algorithm ='auto', leaf_size=30, metric='euclidean',
          metric_params =None, n_jobs=None, n_neighbors=8, p=2,
           weights ='uniform')

Confusion matrix is as follows
  [ [13   0   0]
    [0   15   1]
    [0   0   9] ]
Accuracy metrics

            prediction    recall    f1-score    support

     0       1.00         1.00       1.00        13
     1       1.00         0.94       0.97        16
     2       0.90         1.00       0.95        9
    accuracy                           0.97      38
   macro avg       0.97      0.98      0.97      38
weighted avg       0.98      0.97      0.97      38

correct prediction   0.9736842105263158
correct prediction   0.02631578497368148.
```