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 Neighbor Algorithm

Training algorithm:

- For each training example (x, f (x)), add the example to the list training examples

Classification algorithm:

- Given a query instance x_q to be classified,
  - Let $x_1 \ldots x_k$ denote the k instances from training examples that are nearest to x_q
  - Return

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

  - Where, $f(x_i)$ function to calculate the mean value of the k nearest training examples.

Data Set:

Iris Plants Dataset: Dataset contains 150 instances (50 in each of three classes)
Number of Attributes: 4 numeric, predictive attributes and the Class

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Program:

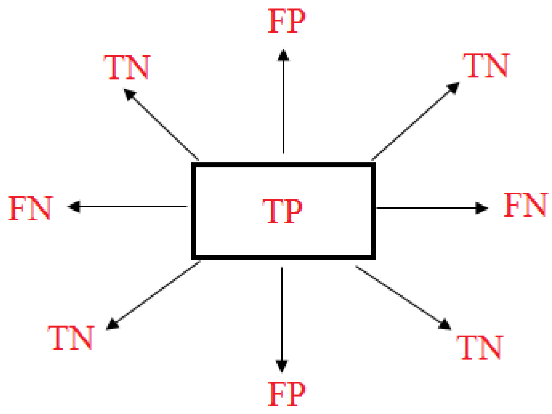
```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import datasets

""" Iris Plants Dataset, dataset contains 150 (50 in each of three
classes)Number of Attributes: 4 numeric, predictive attributes and
the Class
"""
iris=datasets.load_iris()

""" The x variable contains the first four columns of the dataset
(i.e. attributes) while y contains the labels.
"""
x = iris.data
y = iris.target

print ('sepal-length', 'sepal-width', 'petal-length', 'petal-width')
print(x)
print('class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica')
print(y)

""" Splits the dataset into 70% train data and 30% test data. This
means that out of total 150 records, the training set will contain
105 records and the test set contains 45 of those records
"""
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.3)

#To Training the model and Nearest nighbors K=5
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)

#to make predictions on our test data
y_pred=classifier.predict(x_test)

""" For evaluating an algorithm, confusion matrix, precision, recall
and f1 score are the most commonly used metrics.
"""
print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

Output:

```
sepal-length sepal-width petal-length petal-width
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 .   .    .    .    .
 .   .    .    .    .

 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]

class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica
[0 0 0 ………0 0 1 1 1 …………1 1 2 2 2 ………… 2 2]

Confusion Matrix
[[20  0  0]
 [ 0 10  0]
 [ 0  1 14]]

Accuracy Metrics

                Precision  recall    f1-score   support

          0       1.00       1.00      1.00         20
          1       0.91       1.00      0.95         10
          2       1.00       0.93      0.97         15

avg / total       0.98       0.98      0.98         45
```

Basic knowledge

Confusion Matrix



True positives: data points labelled as positive that are actually positive
False positives: data points labelled as positive that are actually negative
True negatives: data points labelled as negative that are actually negative
False negatives: data points labelled as negative that are actually positive

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Actual Positive}}$$

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Predicted Positive}}$$

Accuracy: how often is the classifier correct?

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

F1-Score:

$$\text{F1 Score} = \frac{2.\text{TP}}{2.\text{TP} + \text{FP} + \text{FN}}$$

Support: Total Predicted of Class.

$$\text{Support} = \text{TP} + \text{FN}$$

Example:

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

This is an example confusion matrix for 3 labels: A,B and C

- Now, let us compute **recall** for Label A:
  = TP_A/(TP_A+FN_A)
  = TP_A/(Total Gold for A)
  = TP_A/TotalGoldLabel_A
  = 30/100
  = 0.3
- Now, let us compute **precision** for Label A:
  = TP_A/(TP_A+FP_A)
  = TP_A/(Total predicted as A)
  = TP_A/TotalPredicted_A
  = 30/60
  = 0.5
- Now, let us compute **F1-score** for Label A:

$$\text{F1 Score} = \frac{2.TP}{2.TP + FP + FN}$$

= 2*30 / (2*30 + 60 + 100)

= 0.27

- Support _ A = TP_A + FN_A
  = 30 + (20 + 10)
  = 60