AIM - 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.

KNN (K-Nearest Neighbor) is a simple supervised classification algorithm we can use to assign a class to new data point. It can be used for regression as well, KNN does not make any assumptions on the data distribution, hence it is non-parametric. It keeps all the training data to make future predictions by computing the similarity between an input sample and each training instance.

Below example shows implementation of KNN on iris dataset using scikit-learn library.

Iris dataset has 50 samples for each different species of Iris flower(total of 150). For each sample we have sepal length, width and petal length and width and a species name(class/label).

-> 150 observations

-> 4 features(sepal length, sepal width, petal length, petal width)

-> Classification problem since response is categorical.

Our task is to build a KNN model which classifies the new species based on the sepal and petal measurements. Iris dataset is available in scikit-learn and we can make use of it to build our KNN.

Load the Iris data set and check the features

In [60]:
```
#Import the load_iris function from datsets module
from sklearn.datasets import load_iris
```

In [61]:
```
#Create bunch object containing iris dataset and its attributes.
iris = load_iris()
```

In [62]:
```
type(iris)
```

Out[62]:
```
sklearn.utils.Bunch
```

In [63]:
```
#Print the iris data
iris.data
```

Out[63]:
```
array([[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],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

In [64]:
```
#Names of 4 features (column names)
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

In [65]:
```
#Integers representing the species: 0 = setosa, 1=versicolor, 2=virginica
print(iris.target)
```

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

In [66]:
```
# 3 classes of target
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [67]:
```
# Feature matrix in a object named X
X = iris.data
# response vector in a object named y
y = iris.target
```

Train the Model

In [68]:
```
# splitting the data into training and test sets (80:20)
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_sta
```

In [69]:
```
#import the KNeighborsClassifier class from sklearn and fit the model
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X,y)
```

Out[69]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [70]:
```
# making prediction from X_test data
y_pred=knn.predict(X_test)
```

Printing confusion matrix and score

In [71]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[71]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

In [72]:
```
# importing metrics model to check the accuracy
from sklearn import metrics
metrics.accuracy_score(y_test,y_pred)
```

Out[72]:
```
1.0
```

Testing the model in our own data

In [73]:
```
#0 = setosa, 1=versicolor, 2=virginica
classes = {0:'setosa',1:'versicolor',2:'virginica'}

#Making prediction on some unseen data
#predict for the below two random observations
x_new = [[3,4,5,2],
         [5,4,2,2]]
y_predict = knn.predict(x_new)

print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

```
versicolor
setosa
```

In [ ]: