

Machine Learning
Course Code: BCSE3093
ETE-Experiment Report
For
Bachelor Of
Engineering & Technology



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
UTTAR PRADESH

Student Name: Krishna Gopal

Admission No: 18SCSE1010489

Semester : V / FALL 2020/21

Aim:-

Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions.

Theory:-

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

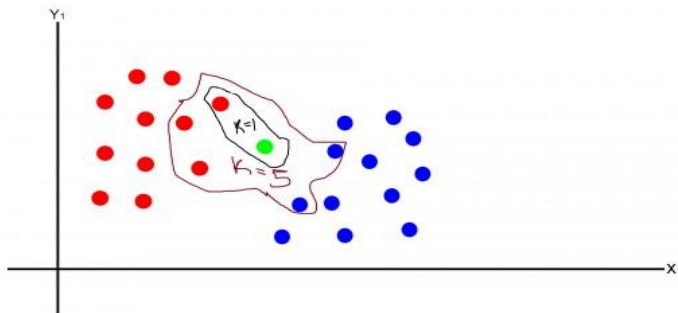
What happens in KNN, we trained the model and after that we want to test our model , means we want to classify our new data (test-data), for that we will check **some (K)** classes around it and assign the most common class to the test-data.

K- Number of nearest neighbors

K=1 means the testing data are given the same level as the closet example in training set.

K=4 means the labels of the four closet classes are check and most common class is assign to the testing data.

How does KNN is work ?

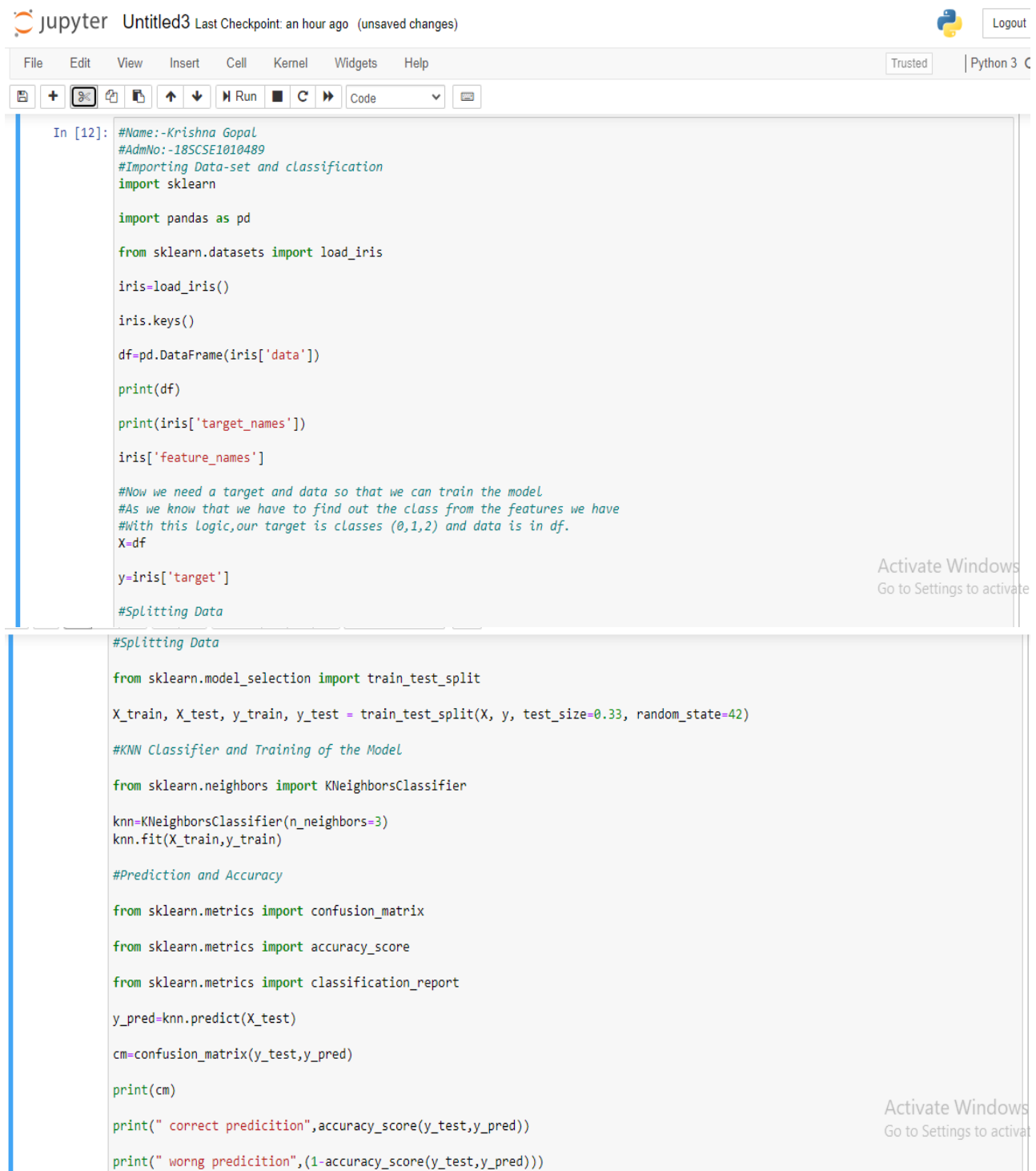


Let's understand it with the above given diagram

1. In this diagram we have 2 classes one blue class one red class
2. Now we have a new green point, we have to find out whether this point is in class red or blue
3. For this, we will define the value of K
4. At K= 1, we will see the distance from the green point to the nearest points, and select the point with lowest distance and classify the green point in that class, here it red.
5. At K=5 We will calculate the distance from the green point to the nearest points and select the five points with the lowest distance and classify the green point to the most common class, that is red here.
6. **How to choose the value of K?** The value of k is not defined, it depends on the cases.

Implementation:-

Data: the iris data: This data consist of total 150 instances (samples) , 4 features , and three classes (targets).



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code is written in Python and is divided into two sections by a horizontal line. The first section, labeled 'In [12]:', contains code for loading the Iris dataset, displaying its structure, and splitting it into training and testing sets. The second section contains code for training a K-Nearest Neighbors (K-NN) classifier, making predictions on the test set, and evaluating the model's performance using a confusion matrix and accuracy score. Comments in the code provide context for each step.

```
In [12]: #Name:-Krishna Gopal
#AdmNo:-18SCSE1010489
#Importing Data-set and classification
import sklearn

import pandas as pd

from sklearn.datasets import load_iris

iris=load_iris()

iris.keys()

df=pd.DataFrame(iris['data'])

print(df)

print(iris['target_names'])

iris['feature_names']

#Now we need a target and data so that we can train the model
#As we know that we have to find out the class from the features we have
#With this logic,our target is classes (0,1,2) and data is in df.
X=df

y=iris['target']

#Splitting Data

#Splitting Data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

#KNN Classifier and Training of the Model

from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)

#Prediction and Accuracy

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

y_pred=knn.predict(X_test)

cm=confusion_matrix(y_test,y_pred)

print(cm)

print(" correct prediction",accuracy_score(y_test,y_pred))

print(" wrong prediction",(1-accuracy_score(y_test,y_pred)))
```

```
print(" wrong prediction",(1-accuracy_score(y_test,y_pred)))
```

```
      0    1    2    3
0  5.1  3.5  1.4  0.2
1  4.9  3.0  1.4  0.2
2  4.7  3.2  1.3  0.2
3  4.6  3.1  1.5  0.2
4  5.0  3.6  1.4  0.2
..  ...  ...  ...  ...
145 6.7  3.0  5.2  2.3
146 6.3  2.5  5.0  1.9
147 6.5  3.0  5.2  2.0
148 6.2  3.4  5.4  2.3
149 5.9  3.0  5.1  1.8
```

```
[150 rows x 4 columns]
['setosa' 'versicolor' 'virginica']
[[19  0  0]
 [ 0 15  0]
 [ 0  1 15]]
correct prediction 0.98
wrong prediction 0.020000000000000018
```

Conclusion:-

As we can see only one prediction is wrong , and also our accuracy is 0.98 (98%). The model gave really good results.

Results:-All codes have been executed properly and successfully . The required outputs have been shown in this experiment report file .