**LAB EXERCISE –8**

**k-Nearest Neighbor Algorithm**

1. **Aim of the Experiment:**

Implement and demonstrate k-Nearest Neighbor algorithm. Read the training data from a .CSV file and build the model to classify a test sample. Print both correct and wrong predictions.

1. **Reference to Text book for Algorithms:**

Refer to Section 4.2 in Chapter 4 Similarity Based Learning to understand the working of the algorithm.

**Listing 1:**

Sample Dataset Used: Table 4.2 in that chapter.

**Table 4.2:** Training Dataset T

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | CGPA | Assessment | Project  Submitted | Result |
| 1. | 9.2 | 85 | 8 | Pass |
| 2. | 8 | 80 | 7 | Pass |
| 3. | 8.5 | 81 | 8 | Pass |
| 4. | 6 | 45 | 5 | Fail |
| 5. | 6.5 | 50 | 4 | Fail |
| 6. | 8.2 | 72 | 7 | Pass |
| 7. | 5.8 | 38 | 5 | Fail |
| 8. | 8.9 | 91 | 9 | Pass |

**3. Python Program with Explanation:**

1. Import the function train\_test\_split to split the dataset into training dataset and test dataset.

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

2. Import KNeighborsClassifier model from sklearn.neighbors.

from sklearn.neighbors import KNeighborsClassifier

1. Import classification\_report and confusion\_matrix from sklearn.metrics to measure the quality of predictions.

from sklearn.metrics import classification\_report, confusion\_matrix

4. Import StandardScaler to apply scaling transformations.

from sklearn.preprocessing import StandardScaler

5. Load data from a CSV file into a [Pandas DataFrame](about:blank).

dataset = pd.read\_csv("knearest csv.csv")

6. Print the total number of records and the number of attributes in the dataframe using the function shape().

print(dataset.shape)

7. Print the first 5 records using the function head().

print(dataset.head())

8. The function tail(2) will print the last 2 records in the data frame.

print(dataset.tail(2))

9. Use iloc property to select by position.

Select the columns until (excluding) the last column. Then print it.

X = dataset.iloc[:, :-1].values

Select the fourth column and then print it.

y = dataset.iloc[:, 3].values

print(X)

print(y)

10. Split the dataset into training dataset and test dataset by using the function train\_test\_split().

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

11. Create a new instance of StandardScaler and then fit and transform the scaler to X\_train and X\_test. Standardize X\_train and X\_test by computing the mean and standard deviation by the transform() function on a training set so as to later reapply the same transformation on the testing set.

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

12. Use KNeighborsClassifier model. The number of neighbours ‘k’ to consider is given through the parameter n\_neighbors.

classifier = KNeighborsClassifier(n\_neighbors=3)

13. The model takes as input two arrays: an array X\_train, holding the training instances, and an array y\_train holding the class labels for the training instances. Then train the classifier using the function fit().

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

14. To make predictions, the predict method of the KNeighborsClassifier class is used.

y\_pred = classifier.predict(X\_test)

15. Generate classification report & confusion matrix to measure the quality of predictions.

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

17. After training, the fitted model can be used to predict a new instance.

The new instance tested is [ CGPA:9.1, Assessment:85, Project Submitted:8 ]

print([9.1,85,8.])

predicted = classifier.predict([[9.1,85,8.]])

print(predicted)

**4. Complete Program:**

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.preprocessing import StandardScaler

dataset = pd.read\_csv("knearest csv.csv")

print(dataset.shape)

print(dataset.head())

print(dataset.tail(2))

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 3].values

print(X)

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

classifier = KNeighborsClassifier(n\_neighbors=3)

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

y\_pred = classifier.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

print([9.1,85,8.])

predicted = classifier.predict([[9.1,85,8.]])

print(predicted)

**Output:**

(8, 4)

CGPA Assessment Project Result

0 9.2 85 8 Pass

1 8.0 80 7 Pass

2 8.5 81 8 Pass

3 6.0 45 5 Fail

4 6.5 50 4 Fail

CGPA Assessment Project Result

6 5.8 38 5 Fail

7 8.9 91 9 Pass

[[ 9.2 85. 8. ]

[ 8. 80. 7. ]

[ 8.5 81. 8. ]

[ 6. 45. 5. ]

[ 6.5 50. 4. ]

[ 8.2 72. 7. ]

[ 5.8 38. 5. ]

[ 8.9 91. 9. ]]

['Pass' 'Pass' 'Pass' 'Fail' 'Fail' 'Pass' 'Fail' 'Pass']

[[1 0]

[0 1]]

precision recall f1-score support

Fail 1.00 1.00 1.00 1

Pass 1.00 1.00 1.00 1

accuracy 1.00 2

macro avg 1.00 1.00 1.00 2

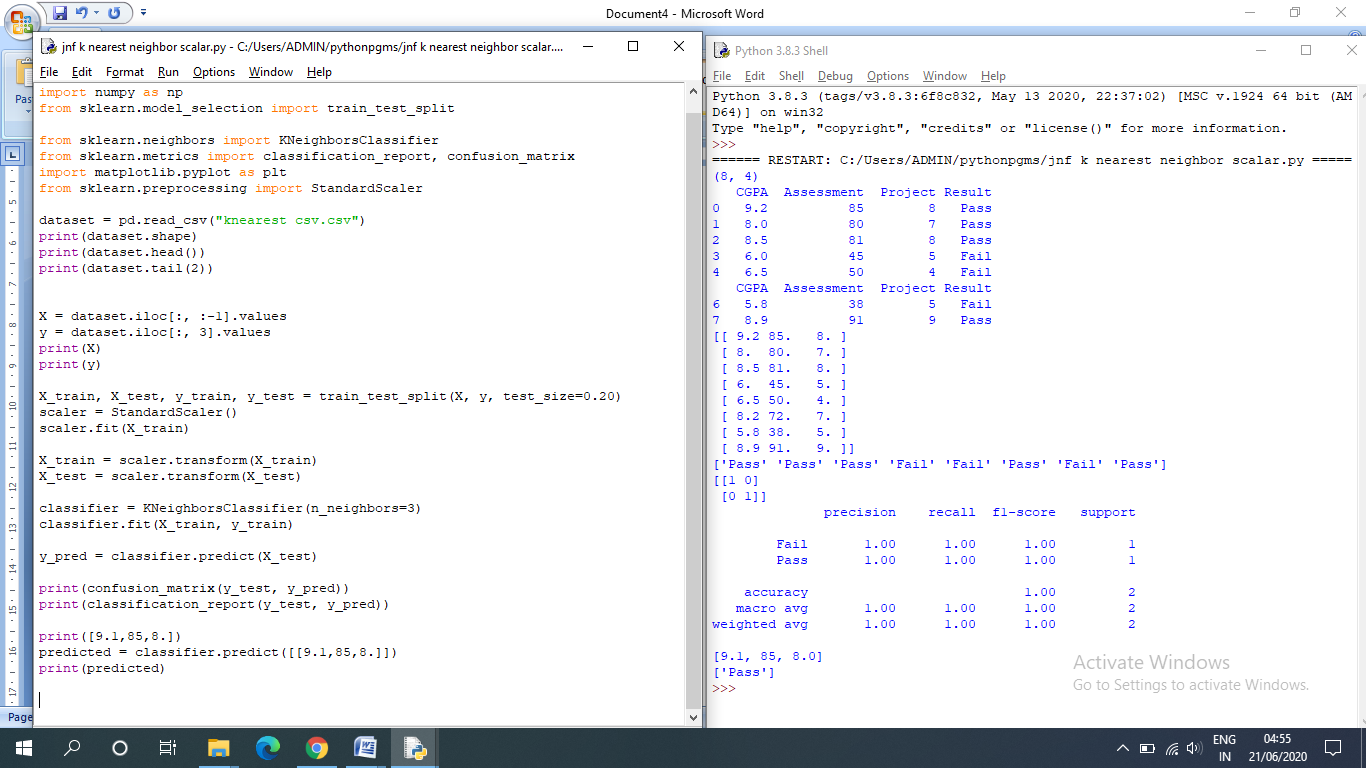
weighted avg 1.00 1.00 1.00 2

[9.1, 85, 8.0]

['Pass']

>>>

**Screenshot of the Output:**



**Listing 2:**

import pandas as pd

import numpy as np

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

import matplotlib.pyplot as plt

# Load the Iris data set

dataset = pd.read\_csv("Iris.csv")

print(dataset.shape)

print(dataset.head())

print(dataset.tail(2))

print(dataset.describe())

# Split the Iris features into input and output columns

X = dataset.iloc[:, 1:5].values

y = dataset.iloc[:, 5].values

# Split the data matrix into train and test dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

#Train the model using KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=3)

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

y\_pred = classifier.predict(X\_test)

# Evaluate the model and print the confusion matrix and classification report

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

# Print the accuracy score

accuracy = accuracy\_score(y\_test, y\_pred)\*100

print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + ' %.')

**Output:**

Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

= RESTART: C:/Users/ADMIN/pythonpgms/Review/jnf k nearest neighbor scalar iris.py

(150, 6)

Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species

0 1 5.1 3.5 1.4 0.2 Iris-setosa

1 2 4.9 3.0 1.4 0.2 Iris-setosa

2 3 4.7 3.2 1.3 0.2 Iris-setosa

3 4 4.6 3.1 1.5 0.2 Iris-setosa

4 5 5.0 3.6 1.4 0.2 Iris-setosa

Id SepalLengthCm ... PetalWidthCm Species

148 149 6.2 ... 2.3 Iris-virginica

149 150 5.9 ... 1.8 Iris-virginica

[2 rows x 6 columns]

Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm

count 150.000000 150.000000 150.000000 150.000000 150.000000

mean 75.500000 5.843333 3.054000 3.758667 1.198667

std 43.445368 0.828066 0.433594 1.764420 0.763161

min 1.000000 4.300000 2.000000 1.000000 0.100000

25% 38.250000 5.100000 2.800000 1.600000 0.300000

50% 75.500000 5.800000 3.000000 4.350000 1.300000

75% 112.750000 6.400000 3.300000 5.100000 1.800000

max 150.000000 7.900000 4.400000 6.900000 2.500000

[[11 0 0]

[ 0 9 0]

[ 0 0 10]]

precision recall f1-score support

Iris-setosa 1.00 1.00 1.00 11

Iris-versicolor 1.00 1.00 1.00 9

Iris-virginica 1.00 1.00 1.00 10

accuracy 1.00 30

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30

Accuracy of our model is equal 100.0 %.

>>>

**Screenshot of the Output:**

