**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Department of Electronics and Telecommunication Engineering**

**Subject: Machine Learning Program: B.Tech/MBA.Tech**

**Sem: III/V ACAY: 2020-21**

**EXPERIMENT NO. 6**

**Aim:**

1. To be able to apply K Nearest Neighbor (KNN) Classifier using sklearn.
2. To be able to interpret the results obtained from KNN.

**Software:**  PYTHON.

**Prerequisite:**

|  |  |
| --- | --- |
| Sr. No | Concepts |
| 1. | Knowledge of K Nearest Neighbor Classification |

**Outcome:**

After successful completion of this experiment students will be able to:

1. Implement KNN by using sklearn.
2. Interpret the results obtained from different data sets and predict class of the given data sample.

**Theory:**

* The data which we will be using for KNN is in a iris-data.csv file.
* For loading the data use the command:

df=pd.read\_csv(‘iris\_data.csv’)

**TO BE COMPLETED BY STUDENTS**

* Students must upload the soft copy of the program in the given format.

|  |
| --- |
| Name of the Experiment |
| Roll No.: N049 Name: Tarun Tanmay |
| Program: MBATech CE Semester: 5 |
| Date of Performance:28/8/20 Date of Submission: 20/9/20 |

**Step 1: Importing the Relevant Libraries**

#Experiment 6

#K nearest Neighbour

import pandas as pd

from math import sqrt

### ****Step 2: Loading the Data****

df\_test=pd.read\_csv('test\_knn.csv')

df\_test

### ****Step 3: Visualizing the Data Frame****

**A screenshot of a cell phone

Description automatically generated**

### ****Step 4: Split data set into train and test samples. Normalize train samples****

#Definition for Euclidean Distance between two Points:

def euclidean\_distance(row1,row2):

distance=0.0

ln=len(row1)

for i in range(0,ln-1):

temp1=(row1[i]-row2[i])\*\*2

distance=distance+temp1

euc\_distance=sqrt(distance)

return euc\_distance

test\_data=df\_test.to\_numpy()

#test\_data

row0=test\_data[0]

for row in test\_data:

distance=euclidean\_distance(row0,row)

print(distance)

A screenshot of a cell phone

Description automatically generated

#Definition for Euclidean Distance between two Points:

def euclidean\_distance(row1,row2):

distance=0.0

ln=len(row1)

for i in range(0,ln-1):

temp1=(row1[i]-row2[i])\*\*2

distance=distance+temp1

euc\_distance=sqrt(distance)

return euc\_distance

test\_data=df\_test.to\_numpy()

#test\_data

row0=test\_data[0]

for row in test\_data:

distance=euclidean\_distance(row0,row)

print(distance)

A screenshot of a cell phone

Description automatically generated

def get\_neighbors(train, test\_row, num\_neighbors):

#initialising a list

dist=list()

for train\_row in train:

edist=euclidean\_distance(test\_row,train\_row)

dist.append((train\_row,edist))

dist.sort(key=lambda tup:tup[1])

neighbors=list()

for i in range(num\_neighbors):

neighbors.append(dist[i][0])

return neighbors, dist

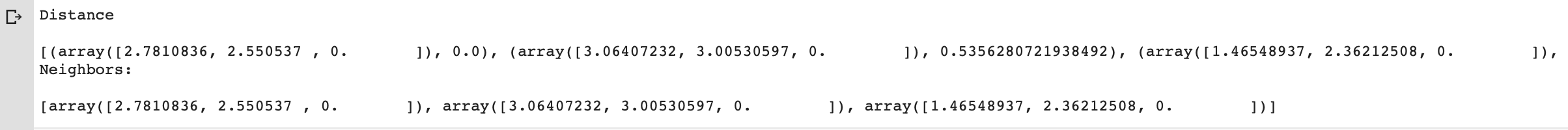
n,d= get\_neighbors(test\_data,test\_data[0],3)

print('Distance \n')

print(d)

print('Neighbors: \n')

print(n)



### ****Step 5: Predict class of the given data for the given value of k****

#Predict based on the majority among neighbors

def predict\_class(train, test\_row,numb\_n):

neig,ecl\_dis=get\_neighbors(train,test\_row,numb\_n)

output=[row[-1] for row in neig]

pred=max(set(output), key=output.count)

return pred

pred\_out=predict\_class(test\_data, test\_data[8],5)

print('Expected Value: ',test\_data[8][-1])

print('Got: ',pred\_out)

A screenshot of a cell phone

Description automatically generated

### ****Step 6: Interpret the results and implement other models****

import seaborn as sn

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

df=pd.read\_csv('iris\_data.csv')

df

### A screenshot of a cell phone Description automatically generated

X=df.iloc[:,:-1]

y=df.iloc[:,4]

X\_train, X\_test, y\_train, y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=1)

#preprocess the data

scaler=StandardScaler()

#scaler.fit converts p to q using q=(p-mean)/std

scaler.fit(X\_train)

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

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

classifier=KNeighborsClassifier(n\_neighbors=5)

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

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

confusion\_matrix=pd.crosstab(y\_test,y\_pred,rownames=['Actual'], colnames=['Predicted'])

sn.heatmap(confusion\_matrix,annot=True)

### A screenshot of a cell phone Description automatically generated

### ****Step 7: Determine test accuracy****

accuracy=27/30

accuracy

### A picture containing table Description automatically generated

### Step 8: Conclusion

### A screenshot of a cell phone Description automatically generated