Practical No. 3

Roll No.: K041	Name: Anish Sudhan Nair
Class: B.Tech Cybersecurity	Batch: K2/A2
Date of Practical: 29/01-05/02 2022	Date of Submission: 05/02/2022
Grade:	

Aim: To implement K nearest neighbour classification algorithm

Prerequisite:

1. Working of KNN classification algorithm
2. Understanding of fundamental programming constructs in C/C++/Java
3. Basic features of WEKA tool

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

- Understand the working of KNN algorithm through implementation and the various measures used for checking its performance
- Use Classifier tab in WEKA, apply KNN classification algorithm and analyse the model created.
- Build KNN classifier using Knowledge Flow in WEKA

Theory:

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time.

In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case.

Classifications are discrete and do not imply order. Continuous, floating-point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm.

The simplest type of classification problem is binary classification. In binary classification, the target attribute has only two possible values: for example, high credit rating or low credit rating. Multiclass targets have more than two values: for example, low, medium, high, or unknown credit rating.

K nearest neighbours is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Algorithm

A tuple is classified by a majority vote of its neighbours, with the case being assigned to the class most common amongst its K nearest neighbours measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbour.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

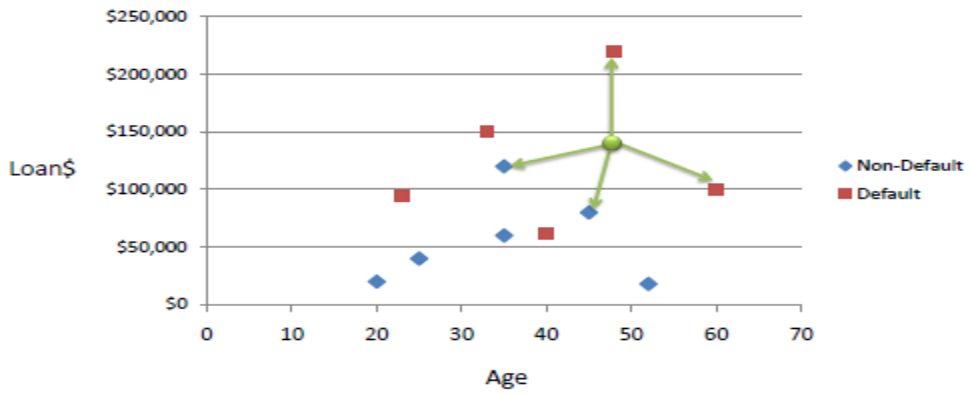
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Example:

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbour is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

With K=3, there are two Default=Y and one Default=N out of three closest neighbours. The prediction for the unknown case is again Default=Y.

****

(TO BE COMPLETED BY STUDENTS)

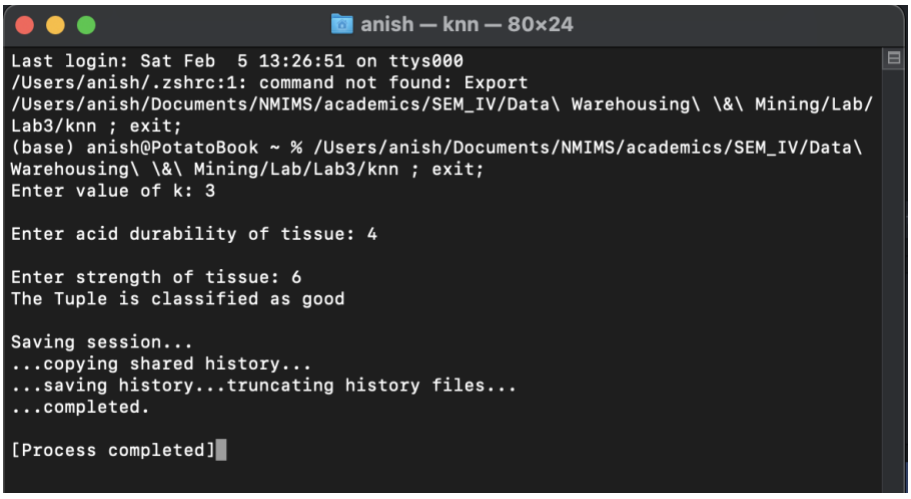
Roll No.: K041	Name: Anish Sudhan Nair
Class: B.Tech Cybersecurity	Batch: K2/A2
Date of Practical: 29/01-05/02 2022	Date of Submission: 05/02/2022
Grade:	

A production unit of a company conducts survey to find out people's opinions about the tissue paper that they manufacture. Based on the questionnaire that was taken, the following data set was created. The factory now wants to test the new tissue paper produced by them with the attribute of acid durability = 3 and strength being 7.

Acid durability	Strength	Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good
2	3	Good
6	3	Bad
2	5	Good
5	4	Bad

a. Preprocess and implement a KNN classifier model with value of k being 3 and use Euclidean as the distance measure.

Output:



Code:

knn.c

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

// store acid, strength and class
// take value of k from user, here 3
//take tuple from user
//calc distance array
//then sort in ascending order
//based on value of k=3, top 3, if k=5, look at top5 vaues
//look at corresponding class
//max one would be the classfying class

//acid,stremght, class, distance -> structure
//array of strcutures

//Swap elements of array (the elements are structures)

struct tissue{
  int acidTissue;
  int strengthTissue;
  char classTissue;
  int distanceTissue;
};

struct tissue tissueArray[8];
int acidArray[8]={7,7,3,1,2,6,2,5};
int strengthArray[8]={7,4,4,4,3,3,5,4};
char class[8]={"b","b","g","g","g","b","g","b"};
int distanceArray[8];


float euclideanDistance(int acid, int strength, int i){

  int x=acid-acidArray[i];
  int y=strength-strengthArray[i];

  float distance=sqrt(pow(x,2)+pow(y,2));

  return distance;
}

int distanceSort(){

  struct tissue Temp;
  for (int i = 0; i < 8; i++) {
    for (int j = i; j < 8; j++) {
      if (tissueArray[j].distanceTissue>tissueArray[i].distanceTissue) {
        Temp=tissueArray[i];
        tissueArray[i]=tissueArray[j];
        tissueArray[i]=Temp;
      }
    }
  }
  return 0;
}

int classifier(int k){

  int badCount=0, goodCount=0;

  for (int i = 0; i < k; i++) {
    if (tissueArray[i].classTissue=="b") {
```

```
64          badCount++;
65        }
66        else {
67          goodCount++;
68        }
69      }
70
71      if (badCount>goodCount) {
72        printf("The Tuple is classified as bad\n");
73      }
74      else {
75        printf("The Tuple is classified as good\n");
76      }
77
78      return 0;
79    }
80
81    int main() {
82
83
84      int k, acid_durability, strength;
85
86      printf("Enter value of k: ");
87      scanf("%d",&k);
88
89      fflush(stdin);
90
91      // printf("Working??");
92
93      //CRIES IN CODE
94
95      printf("\nEnter acid durability of tissue: ");
96      scanf("%d",&acid_durability);
97
98      fflush(stdin);
99
100     printf("\nEnter strength of tissue: ");
101     scanf("%d",&strength);
102
103     fflush(stdin);
104
105
106   //Distance Caclulation
107     for (int i = 0; i < 8; i++) {
108       distanceArray[i]=euclideanDistance(acid_durability, strength, i);
109       tissueArray[i].acidTissue=acidArray[i];
110       tissueArray[i].strengthTissue=strengthArray[i];
111       tissueArray[i].classTissue=class[i];
112       tissueArray[i].distanceTissue=distanceArray[i];
113     }
114
115   //Sorting
116
117     distanceSort();
118
119   //Classifying
120
121     classifier(k);
122
123     return 0;
124   }
125
```

b. Using WEKA tool: For the same data set, build a KNN model and classify the tuple, With attribute of acid durability = 4 and strength being 6 (Supply test data)

Training Dataset

tissue-paper.arff

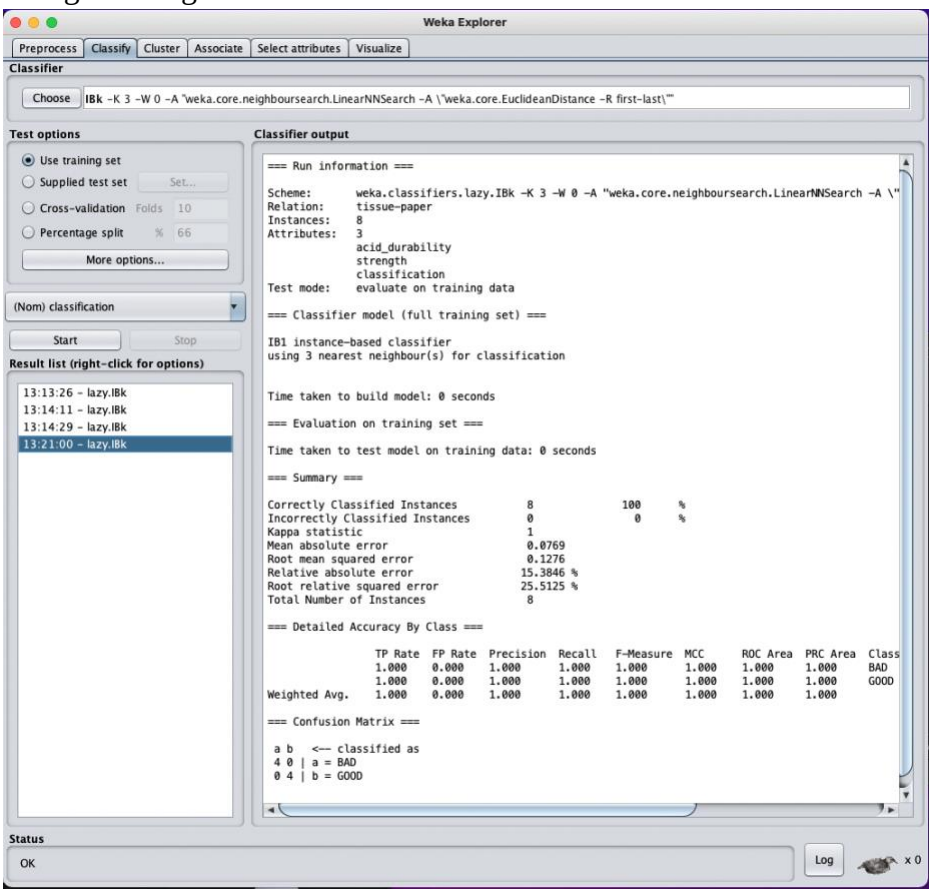
```
@relation tissue-paper

@attribute acid_durability numeric
@attribute strength numeric
@attribute classification {BAD,GOOD}

@data
7,7,BAD
7,4,BAD
3,4,GOOD
1,4,GOOD
2,3,GOOD
6,3,BAD
2,5,GOOD
5,4,BAD
```

Testing Dataset

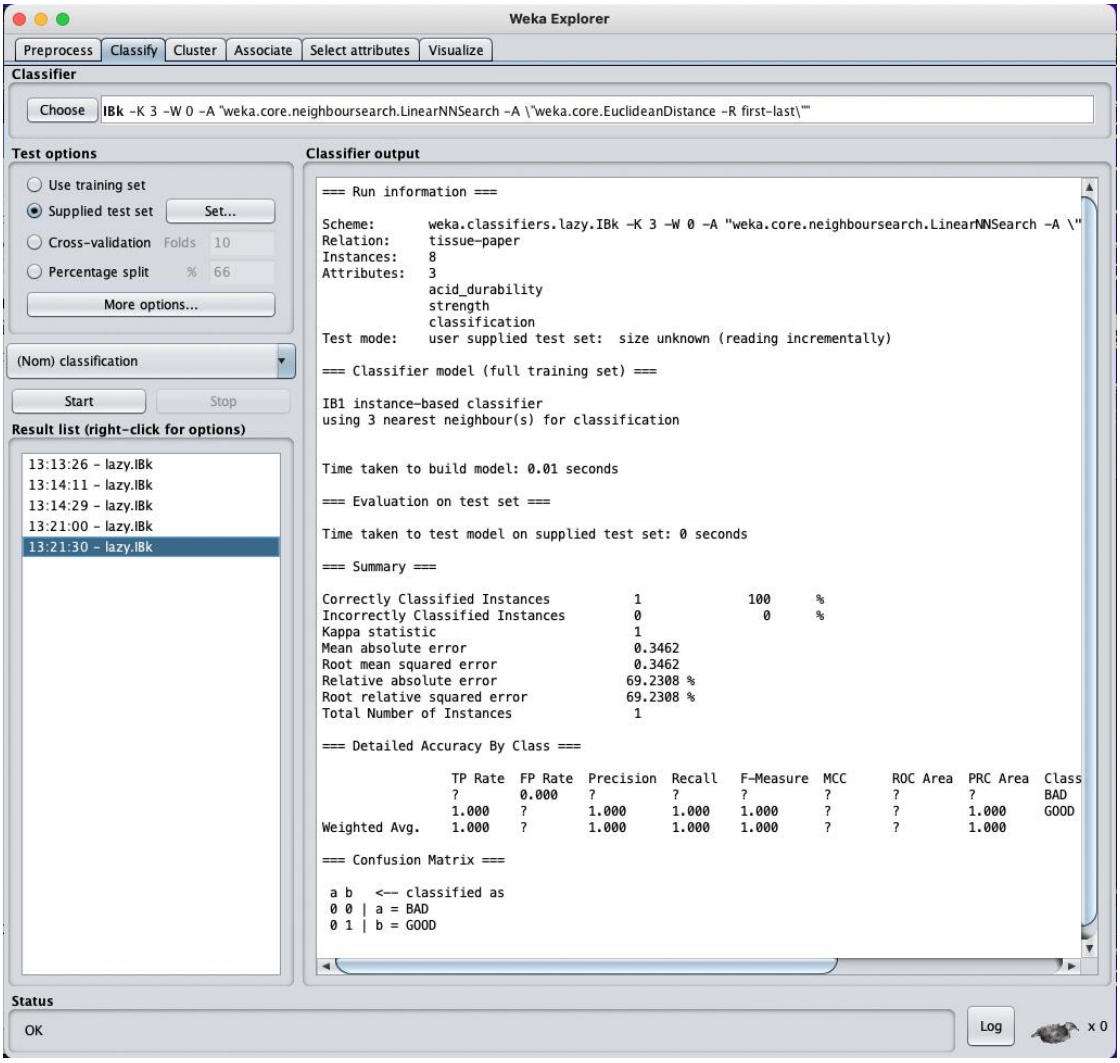
tissue-paper-test.arff

```
@relation tissue-paper

@attribute acid_durability numeric
@attribute strength numeric
@attribute classification {BAD,GOOD}

@data
4,6,GOOD
```

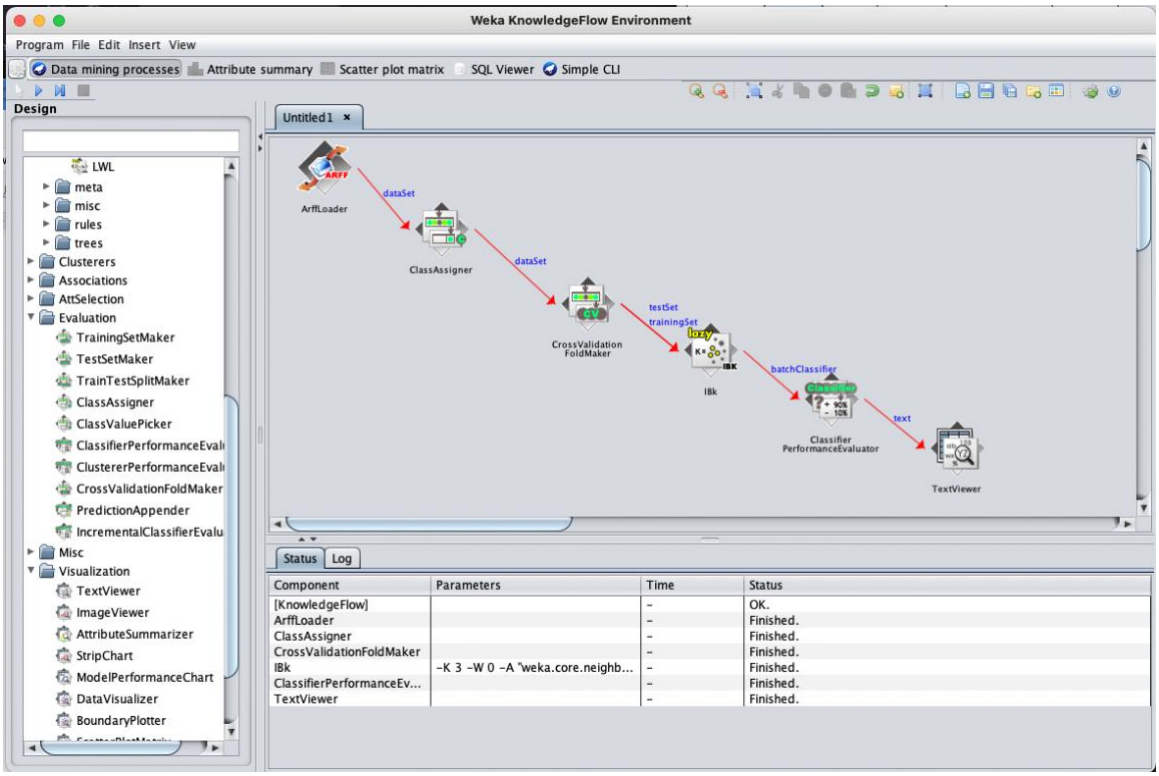
Using Training dataset

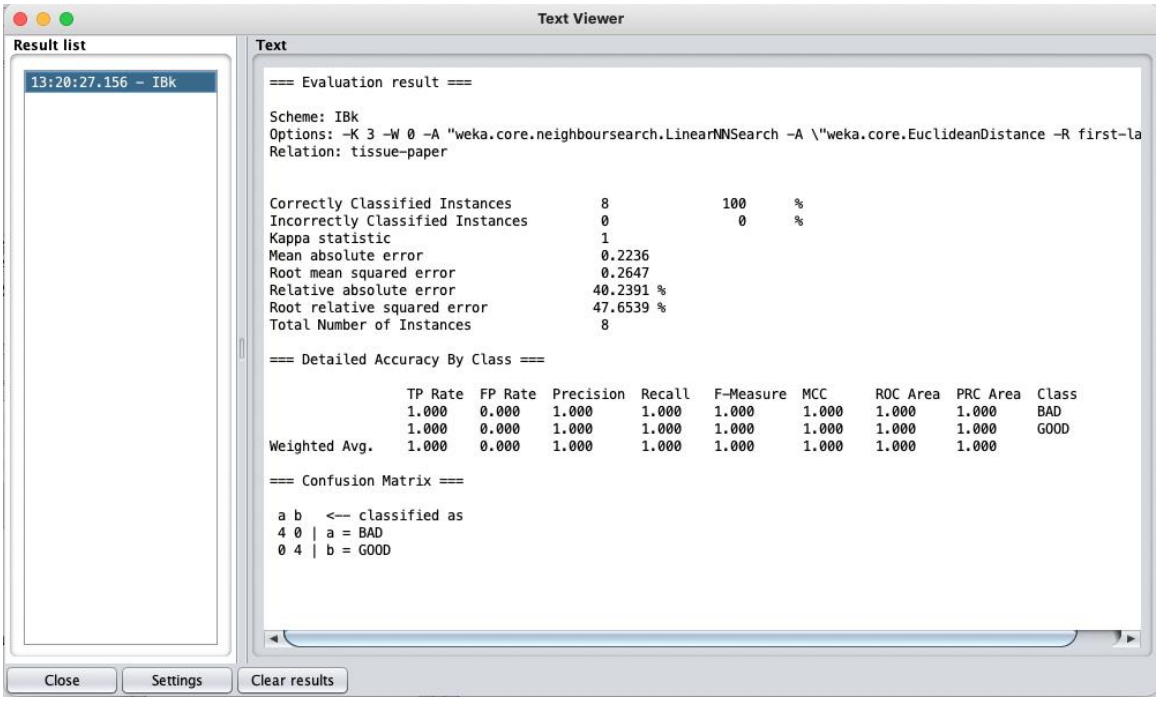


Using Test dataset



c. Use Knowledge flow in WEKA and build KNN classifier for the same data set. (Use cross validation)





Questions to be answered:

1. Why is KNN algorithm also called as lazy learning and Nonparametric learning algorithm?

   ➔ KNN algorithm does not involve a training phase or model fitting wherein it learns to discriminate between data points to make assertions. It instead memorises the entire dataset and every time a new data point is introduced, it scans through the entire dataset looking for the K nearest neighbours. Because of this lack of "training" by the model (or more appropriately, algorithm) it is called a lazy learning algorithm.

   ➔ KNN is a non-parametric learning algorithm since it does not have any parameters for defining the classes for classification and instead scans the dataset for nearest neighbours every time, working under the assumption that the previous tuples have been correctly classified.

****