Assignment: B13

Roll No: 4430

Aim

Implementation of K-NN approach take suitable example.

Problem Statement

Study and implement a K-Nearest Neighbors algorithm using java programming language.

Learning Objectives

1. To understand concepts of Classification.
2. To study K-NN algorithm for classification of data.

Mathematical Model

Let S be the system for implementing Lexical analyzer for sample language using LEX.

S = (s, e, x, y, F, scr, DD, NDD, Mem_{sh}, sucess case, failure case, CPU core count)

Where, s = Start
State e = End
State x = Input
y = Output
F = Set of function
DD = Deterministic Data
NDD = Non-Deterministic Data

Start State =main()

End State = It is distinct end of program.

Input:
Let x be the set of inputs such as $x = \{x_1, x_2\}$ x_1 = Height x_2 = Weight

Output:
Let y be the set of outputs such as , y = Data objects are classified.

F = Set of function
F = { f_1, f_2, f_3 }
f_1 = NearestNeighbour(ArrayList¡DataEntry¿dataset,int k f_2 = Distance(DataEntry a,DataEntry b) f_3 = classify(DataEntry e)

scr = screen

Deterministic Data:

Height and weight attribute for data object.

Non-Deterministic Data:

Input values entered by user for data object.

MEM_{sh} = Shared memory is not used in this case.

Success Case = classification is done succesfully.

Failure Case = Classification is not done sucessfully.

CPU core count =2

Theory

- **Introduction**

  The K-nearest neighbors algorithm is a method for classifying objects based on closest training examples in the feature space.

- **KNN algorithm:**

  K-Nearest Neighbors (KNN) classification divides data into a test set and a training set. For each row of the test set, the K nearest (in Euclidean distance) training set objects are found, and the classification is determined by majority vote with ties broken at random. If there are ties for the K^{th} nearest vector, all candidates are included in the vote.

Program Code and Output

```
import java.io.*; import java.util.ArrayList;
import java.util.HashMap;
/**
  *
     *  An implementation of knn.
     *  Uses Euclidean distance weighted by 1/distance*
     *  Main method to classify if entry is male or female based on:
     *  Height, weight
  */ public class NearestNeighbour{ public static void main(String[] args) throws IOException
{
System.out.println("\nData Set is:\n");
System.out.println("\tHeight\tWeight\tGender");
System.out.println("\t175\t80\tMale");
      System.out.println("\t193.5\t110\tMale");
            System.out.println("\t183\t92.8\tMale");
System.out.println("\t160\t60\tMale");
System.out.println("\t177\t73.1\tMale");
System.out.println("\t170\t80\tFemale");
System.out.println("\t150\t55\tFemale");
System.out.println("\t159\t63.2\tFemale");
System.out.println("\t180\t70\tFemale");
System.out.println("\t163\t110\tFemale");
BufferedReader b=new BufferedReader(new InputStreamReader(System.in));
System.out.println("\nEnter Height of Person in cms:"); int ht=Integer.parseInt(b.readLine());
System.out.println("\nEnter Weight of Person in kgs:"); int wt=Integer.parseInt(b.readLine());
ArrayList<NearestNeighbour.DataEntry> data = new ArrayList<NearestNeighbour.DataEntry>()
```

```
data.add(new DataEntry(new double[]{175,80}, "Male"));
data.add(new DataEntry(new double[]{193.5,110}, "Male"));
data.add(new DataEntry(new double[]{183,92.8}, "Male"));
data.add(new DataEntry(new double[]{160,60}, "Male"));
data.add(new DataEntry(new double[]{177,73.1}, "Male"));
data.add(new DataEntry(new double[]{170,80}, "Female"));
data.add(new DataEntry(new double[]{150,55}, "Female"));
data.add(new DataEntry(new double[]{159,63.2}, "Female"));
data.add(new DataEntry(new double[]{180,70}, "Female"));
data.add(new DataEntry(new double[]{163,110}, "Female"));
NearestNeighbour nn = new NearestNeighbour(data, 3); //3 neighbours
System.out.println("\nClassified as: "+nn.classify(new DataEntry(new double[]{ht, wt},"I
}
private int k; private ArrayList<Object> classes; private
ArrayList<DataEntry> dataSet;
/**
  *
     *  @param dataSet The set
     *  @param k The number of neighbours to use
  */ public NearestNeighbour(ArrayList<DataEntry> dataSet, int k){ this.classes = new
ArrayList<Object>(); this.k = k; this.dataSet = dataSet;
//Load different classes for(DataEntry entry : dataSet){ if(!classes.contains(entry.getY()))
classes.add(entry.getY());
} }
private DataEntry[] getNearestNeighbourType(DataEntry x){ DataEntry[] retur = new
DataEntry[this.k]; double fjernest = Double.MIN_VALUE; int index = 0;
for(DataEntry tse : this.dataSet){ double distance = distance(x,tse); if(retur[retur.length-
1] == null){ //Hvis ikke fyldt int j = 0;
while(j < retur.length){ if(retur[j] == null){ retur[j]
= tse; break;
} j++; } if(distance > fjernest){ index = j;
fjernest = distance;
}
} else{ if(distance < fjernest){ retur[index] = tse; double f = 0.0;
int ind = 0; for(int j = 0; j < retur.length; j++){ double dt =
distance(retur[j],x); if(dt > f){ f = dt; ind = j;
}
} fjernest = f; index =
ind;
}
} } return retur;
}
private static double convertDistance(double d){ return 1.0/d;
}
/**
     *  Computes Euclidean distance
     *  @param a From
     *  @param b To
     *  @return Distance
  */ public static double distance(DataEntry a, DataEntry b){ double distance = 0.0; int length =
a.getX().length; for(int i = 0; i < length; i++){ double t = a.getX()[i]-b.getX()[i]; distance =
distance+t*t;
} return Math.sqrt(distance);
}
/**
  *
     *  @param e Entry to be classifies
     *  @return The class of the most probable class
  */ public Object classify(DataEntry e){
HashMap<Object,Double> classcount = new HashMap<Object,Double>();
DataEntry[] de = this.getNearestNeighbourType(e);
for(int i = 0; i < de.length; i++){
double distance = NearestNeighbour.convertDistance(NearestNeighbour.distance(de[i], e));
if(!classcount.containsKey(de[i].getY())){
```

```
classcount.put(de[i].getY(), distance);
} else{ classcount.put(de[i].getY(), classcount.get(de[i].getY())+distance);
}
}
//Find right choice Object o = null; double max = 0; for(Object
ob : classcount.keySet()){ if(classcount.get(ob) > max){ max =
classcount.get(ob); o = ob;
} }
return o; }
public static class DataEntry{ private double[] x;
private Object y;
public DataEntry(double[] x, Object y){ this.x = x; this.y = y;
}
public double[] getX(){ return this.x;
}
public Object getY(){ return this.y;
}
}
}
```

Output

```
ameeth@ubuntu-16.0.4:~/CL1$ cd Documents/CL-1\(CRB\)/D5_KNN/
ameeth@ubuntu-16.0.4:~/CL1$  javac NearestNeighbour.java
ameeth@ubuntu-16.0.4:~/CL1$ java NearestNeighbour
Data Set is:
Height Weight Gender
175 80 Male
193.5 110 Male
183 92.8 Male 160 60
Male
177 73.1 Male
170 80 Female
150 55 Female
159 63.2 Female
180 70 Female
163 110 Female
Enter Height of Person in cms: 180
Enter Weight of Person in kgs: 100
Classified as: Male
ameeth@ubuntu-16.0.4:~/CL1$ java NearestNeighbour Data Set is:
Height Weight Gender
175 80 Male
193.5 110 Male
183 92.8 Male 160 60
Male
177 73.1 Male
170 80 Female
150 55 Female
159 63.2 Female
180 70 Female
163 110 Female
Enter Height of Person in cms: 155
Enter Weight of Person in kgs: 20
Classified as: Female
```

Conclusion

We have successfully implemented K-NN algorithm with suitable example.