A  Homework Report on

Anonymization and Diversity

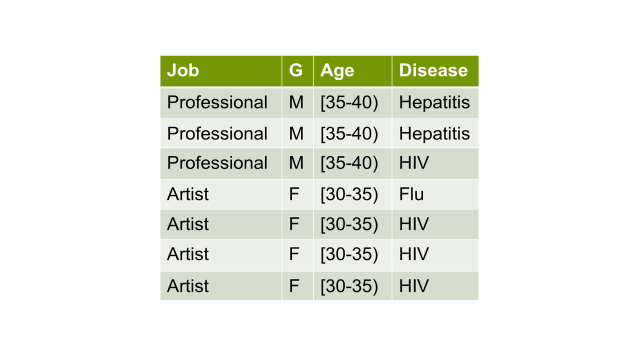
In the partial fulfilment of the academic requirements for

M. S (Cyber Security) under Wright State University

By

Sudarshan Govindarajan               (U00793458)

Dataset used is the one provided in the class room



**Quasi -Identifiers**

* Age
* Zipcode

**Sensitive attributes**

* Disease

**Execution**

* Upon running the program the headers of all the Dataset will be provided.
* Users are request to provide the number of quasi-identifiers.

Ex: 1 or 2

* Users are requested to provide the name of the quasi-identifiers from the Dataset headers provided initially

Eg: Age or Zip code

* Output of Anonymity value is provided
* Sensitive values of all the equivalence class with Diversity of individual class provided
* Diversity value is provided

**Procedure Implemented**

**Anonymity**

* A new Dataset with quasi-identiers is created.
* Using quasi-identifier data set rows that are unique are created

Eg:

Professional, 35-40

Artist, 30-35

* Then number of count for each of the unique quasi-identifier rows is calculated. The value of the least will be considered as anonymity.

**Diversity**

* A new Dataset with quasi-identifiers and sensitive values are created
* Dataset will be divided into the equivalence class. Diversity for individual equivalence class is calculated.
* Diversity of all the individual equivalence class is checked for the consistency.
* Diversity output is provided.

**Implementation of the System**

**package** Anonymousdiversity;

**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.\*;

**public** **class** Anonymous3 {

/\*\*\*

\* Declartion of variables used for Anonymity

\*

\*/

//ArrayList Dataset is represents the input File DataSet

**static** ArrayList<ArrayList<String>> *Dataset*=**new** ArrayList<ArrayList<String>>();

**static** ArrayList<String> *row*;

//quasiDataset represents the Dataset created based on quasi-identifiers.

**static** ArrayList<ArrayList<String>> *quasiDataset*=**new** ArrayList<ArrayList<String>>();

**static** ArrayList<String> *quasirow*;

//Stores the Anonymity value

**static** **int** *Anonymityvalue*;

//Array to store the individual values of a line present in the file

**static** String[] *Data*;

//Array to store the individual values of a line present in the QuasiDataset

**static** String[] *quasiDatasplit*;

//ArrayList stores the sensitive values

**static** ArrayList<String> *sensitivearraylist*[];

**static** **int** *i*=0;

**static** **int** *frequency*;

//To Store the anonymity value

**static** Object *Anonymity*;

/\*\*\*

\* Declartion of variables used for Diversity

\*

\*/

//Individual row from the QuasiDataset

**static** ArrayList *AnonymityList*;

//Conversion of row into array of String Data

**static** String[] *AnonymityDataString*;

**static** Object *DataDiversity*;

//Array to store the individual values of a row present in the Dataset used to identify Diversity

**static** String[] *DiversityDatasplit*;

//Individual row of a Dataset required stored as ArrayList

**static** ArrayList<String> *Diversityrow*;

//Dataset for Diversity

**static** ArrayList<ArrayList<String>> *DiversityDataset*=**new** ArrayList<ArrayList<String>>();

//To obtain number of quasi identifiers from the users

**static** Scanner *user\_input*=**new** Scanner(System.***in***);

**static** **int**[] *Columnnumber*;

**static** Set<ArrayList> *Uniqueset*;

**static** String[] *quasirowvalues*;

**static** **int** *p*=0;

**static** **int**[] *individualblockdiverse*;

**static** **int** *diversityvalue*;

**public** **static** **void** main(String args[])

{

// Reads a Dataset

String fileName = "K-anonymous1.txt";

String line = **null**;

**try** {

// FileReader reads text files in the default encoding.

FileReader fileReader =

**new** FileReader(fileName);

// Always wrap FileReader in BufferedReader.

BufferedReader bufferedReader =

**new** BufferedReader(fileReader);

**int** i=0;

**while**((line = bufferedReader.readLine()) != **null**) {

*row*=**new** ArrayList();

*Data*=line.split(",");

**for**(i=0;i<*Data*.length;i++)

{

*row*.add(*Data*[i]);

}

*Dataset*.add(*row*);

}

// Always close files.

bufferedReader.close();

}

**catch**(FileNotFoundException ex) {

System.***out***.println(

"Unable to open file '" +

fileName + "'");

}

**catch**(IOException ex) {

System.***out***.println(

"Error reading file '"

+ fileName + "'");

// Or we could just do this:

// ex.printStackTrace();

}

//Functions for anonymous and diversity

*anonymous*();

*diversity*();

}

// Function to calculate the anonymity

**private** **static** **void** anonymous() {

System.***out***.println("The data that will be published is the following "+*Dataset*.get(0));

System.***out***.println("Enter the number of Quasi -Identifiers to be provided ");

**int** value= Integer.*parseInt*(*user\_input*.next());

String Identifier[]=**new** String[value];

**for**(**int** i=0;i<value;i++)

{

System.***out***.println("Enter the name of quasi identifier from the Dataset Provided above");

Identifier[i]=*user\_input*.next();

}

// Based on the Quasi identifiers provided above determining the column number of the quasi identifier present in Dataset.

String StringColumn=*Dataset*.get(0).toString();

String Columntoken =StringColumn.substring(1, StringColumn.length()-1);

String Columnsplit[]= Columntoken.split(",");

**int** k;

**for**(k=0;k<Columnsplit.length;k++)

{

Columnsplit[k]=Columnsplit[k].trim();

}

*Columnnumber*=**new** **int**[Identifier.length];

**int** j;

**for**(k=0;k<Columnsplit.length;k++)

{

**for**(j=0;j<Identifier.length;j++)

{

**if**(Columnsplit[k].equalsIgnoreCase(Identifier[j]))

{

*Columnnumber*[j]=k;

}

}

}

//Removing the headers present in the Dataset for smooth operation

*Dataset*.remove(0);

// Creating a Dataset based on values of Quasi-Identifer

Iterator itr=*Dataset*.iterator();

**while**(itr.hasNext())

{

*quasirow*=**new** ArrayList();

String Data= itr.next().toString();

String quasiData=Data.substring(1, Data.length()-1);

*quasiDatasplit*=quasiData.split(",");

//System.out.println(quasiData);

**for**(**int** i=0;i<*quasiDatasplit*.length;i++)

{

**for**(**int** m=0;m<*Columnnumber*.length;m++)

{

**if**(*Columnnumber*[m] == i)

{

*quasirow*.add(*quasiDatasplit*[i]);

}

}

}

*quasiDataset*.add(*quasirow*);

}

/\* Printing of Quasi Dataset

Iterator itr1=quasiDataset.iterator();

while(itr1.hasNext())

{

System.out.println(itr1.next());

}

\*/

//Identify the Unique rows present in QuasiDataset Arraylist

*Uniqueset*=**new** HashSet<>(*quasiDataset*);

System.***out***.println();

Iterator itr2=*Uniqueset*.iterator();

**int**[] Countvalue= **new** **int**[*Uniqueset*.size()];

**int** n=0;

//Checking the Frequency Count of unique row of QuasiDataset

**while**(itr2.hasNext())

{

Countvalue[n]=Collections.*frequency*(*quasiDataset*,itr2.next());

n++;

}

//Finding the smallest value from the count of all the individual unique rows.

*Anonymityvalue*=Countvalue[0];

**for** (**int** p=0;p<n;p++)

{

**if**(Countvalue[p]<*Anonymityvalue*)

{

*Anonymityvalue*=Countvalue[p];

}

}

System.***out***.println("Anonymity value is"+*Anonymityvalue*);

}

**private** **static** **void** diversity() {

Iterator itr3=*Uniqueset*.iterator();

// Creating the Dataset containing the quasi identifier values with the rows whose anonymity value is determined above.

*quasirowvalues*=**new** String[*Uniqueset*.size()];

**while**(itr3.hasNext())

{

*Anonymity*=itr3.next();

*AnonymityList*=(ArrayList)*Anonymity*;

// System.out.println(AnonymityList);

*AnonymityDataString*= **new** String[*AnonymityList*.size()];

*AnonymityDataString*=(String[])*AnonymityList*.toArray(*AnonymityDataString*);

//All the individual values of a quasirow is stored as single string

StringBuilder builder2=**new** StringBuilder();

**for**(String s: *AnonymityDataString*)

{

builder2.append(s);

}

*quasirowvalues*[*p*]=builder2.toString();

*p*++;

}

/\*

for(String s:quasirowvalues)

{

System.out.println(s);

}

\*/

//From the initial Dataset add all the sensitive values for a particular quasi row

//Create a new Dataset with quasi identifier values and sensitive values

Iterator itr4=*Dataset*.iterator();

**while**(itr4.hasNext())

{

*Diversityrow*=**new** ArrayList();

String Diversity= itr4.next().toString();

String DiversityData=Diversity.substring(1, Diversity.length()-1);

*DiversityDatasplit*=DiversityData.split(",");

//System.out.println(quasiData);

**for**(**int** i=0;i<*DiversityDatasplit*.length;i++)

{

**for**(**int** m=0;m<*Columnnumber*.length;m++)

{

**if**(*Columnnumber*[m] == i)

{

*Diversityrow*.add(*DiversityDatasplit*[i]);

}

}

}

//Add row with sensitive values

*Diversityrow*.add(*DiversityDatasplit*[*DiversityDatasplit*.length-1]);

//Addition of each row to create a complete Dataset

*DiversityDataset*.add(*Diversityrow*);

}

//For Each row of a Diversity Dataset calculate the sensitive values.

ArrayList Diversity;

String[] Diversityarray;

**int** t=0;

String partialattribute;

String attribute;

String parsedString;

StringBuilder builder1=**new** StringBuilder();

**int** Diversityarraylength;

**int** AnonymityDatastringlength;

String[] attributesplit;

**boolean** found;

*sensitivearraylist*=**new** ArrayList[*p*];

**for**(*p*=0;*p*<*quasirowvalues*.length;*p*++)

{

*sensitivearraylist*[*p*]=**new** ArrayList<String>();

partialattribute=*quasirowvalues*[*p*];

// System.out.println(partialattribute);

Iterator itr5=*DiversityDataset*.iterator();

**while**(itr5.hasNext())

{

Diversity=(ArrayList) itr5.next();

Diversityarray=**new** String[Diversity.size()];

Diversityarray=(String[]) Diversity.toArray(Diversityarray);

**for**(String s: Diversityarray)

{

builder1.append(s);

}

//System.out.println(builder1.toString());

attribute=builder1.toString();

Diversityarraylength=builder1.length();

**if**( attribute.contains(partialattribute))

{

attributesplit=attribute.split(" ");

*sensitivearraylist*[*p*].add(attributesplit[attributesplit.length-1]);

}

builder1.setLength(0);

}

}

*individualblockdiverse*=**new** **int**[*sensitivearraylist*.length];

//Diversity for each Equivalence class

**for**(*p*=0;*p*<*sensitivearraylist*.length;*p*++)

{

System.***out***.println(*sensitivearraylist*[*p*]);

Set<String> Uniquesensitive=**new** HashSet<>(*sensitivearraylist*[*p*]);

//System.out.println(Uniquesensitive);

//Printing the Diversity value

System.***out***.println("Diversity for individual equivalence class with above sensitive values is :"+ Uniquesensitive.size());

*individualblockdiverse*[*p*]=Uniquesensitive.size();

}

//Checking the diversity values for all the Equivalence class to be similar

*diversityvalue*=*individualblockdiverse*[0];

**if**(*checkdiverse*())

{

System.***out***.println("Diversity for the table is:"+*diversityvalue*);

}

}

**private** **static** **boolean** checkdiverse() {

**for**(**int** i=0; i<*individualblockdiverse*.length; i++){

**if**(*individualblockdiverse*[0] == *individualblockdiverse*[i]){

**return** **true**;

}

}

**return** **false**;

}

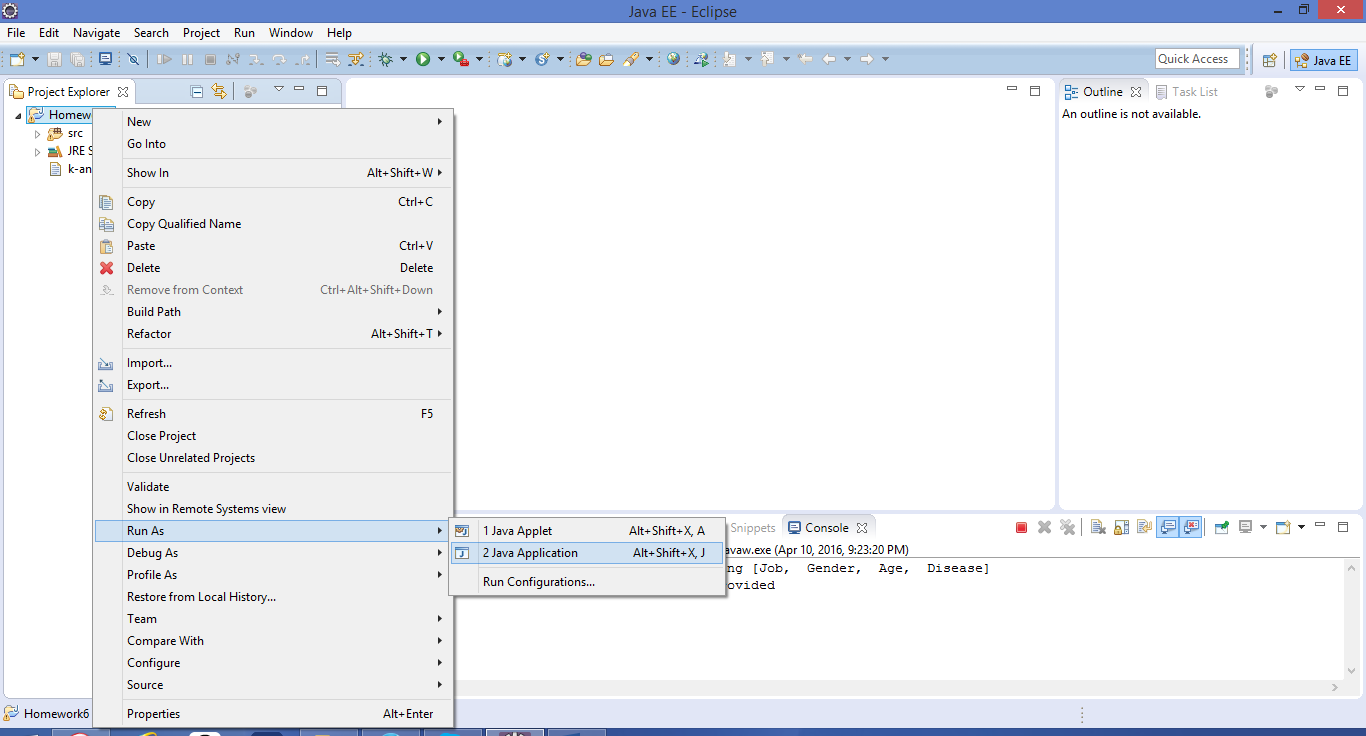
}

**Execution Steps:**

**Start Eclipse**

**File->Import->General->ExistingProjectsintoWorkSpace->Homework6**

**Right click->Run As-> Java Application**

****

**Output:**

