DATA MINING AND DATA WAREHOUSING

NAME : ROOPA

ROLL NUM :175214130.

INDEX….

:

- **<u>PROBLEM DESCRIPTION</u>**

The main goal of the project is to implement the frequent item set Mining algorithm. Frequent item-set mining is a widely used data mining technique used for discovering sets of frequently occurring items in large databases. It speacially used for marketing places.

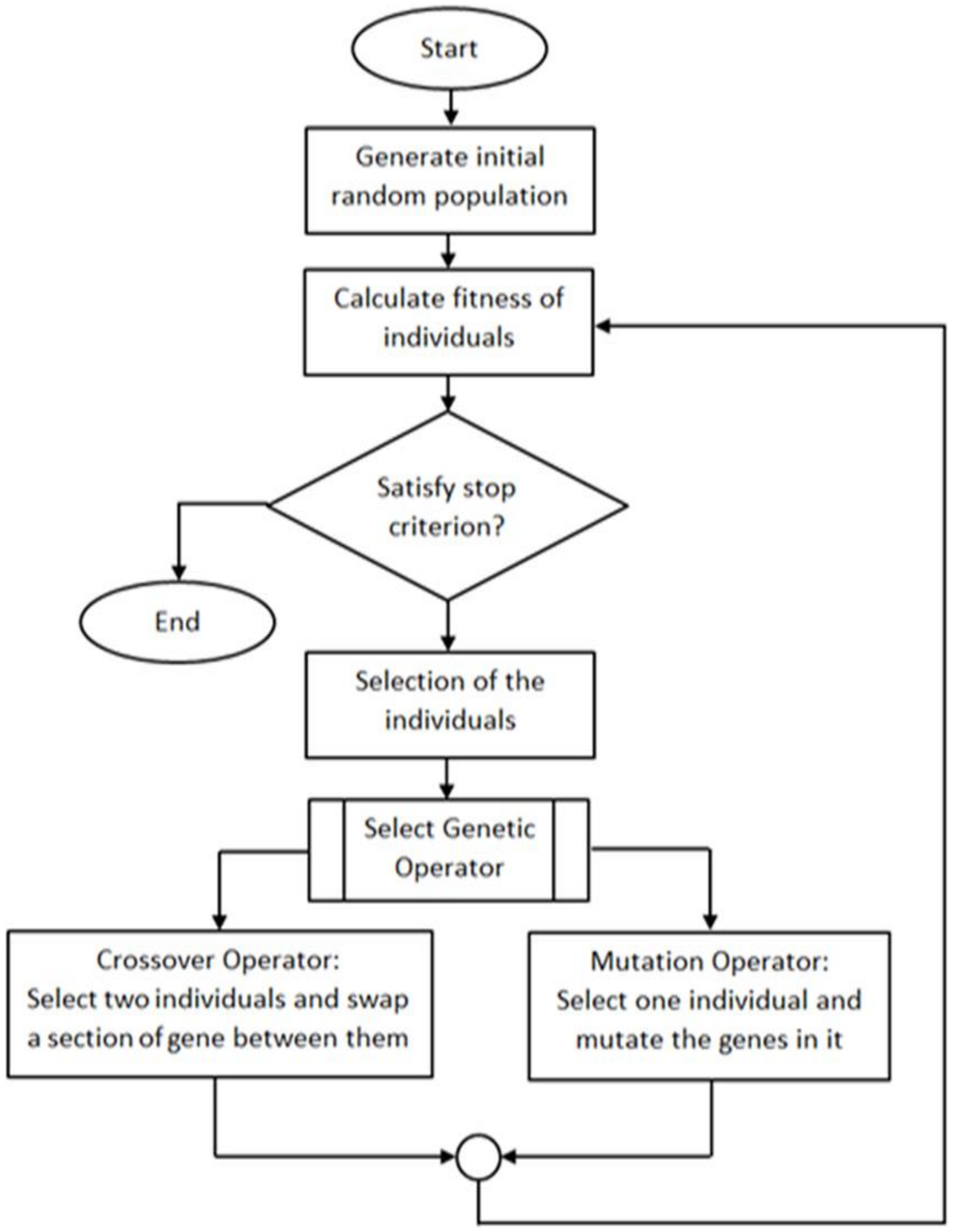
The algorithm to be implemented is the frequent item-set mining algorithm which allows marketing platforms to be able to find probable correlation between their products from the purchase records of their customer.

A very important prominent example of where this application is used is in the amazon website where you attempt to look on a product and on the slide bar. This field of extracting this product correlation this product data is called FIM.

FIM-Frequent Itemset Mining.

The Apriori algorithm was proposed by Agrawal and Srikant in 1994. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing). Each transaction is seen as a set of items (an itemset).

- **ALGORITHM DESIGN**


Start
Generate initial random population
Calculate fitness of individuals
Satisfy stop criterion?
End
Selection of the individuals
Select Genetic Operator
Crossover Operator: Select two individuals and swap a section of gene between them
Mutation Operator: Select one individual and mutate the genes in it

• SOURCE CODE

```
import java.io.*;

import java.util.*;

/** The class encapsulates an implementation of the Apriori algorithm

* to compute frequent itemsets.

*

* Datasets contains integers (>=0) separated by spaces, one transaction by line, e.g.

* 1 2 3

* 0 9

* 1 9

*

* Usage with the command line :

* $ java mining.Apriori fileName support

* $ java mining.Apriori /tmp/data.dat 0.8

* $ java mining.Apriori /tmp/data.dat 0.8 > frequent-itemsets.txt

*

* Usage as library: see {@link ExampleOfClientCodeOfApriori}

* @copyright GNU General Public License v3

* No reproduction in whole or part without maintaining this copyright notice

* and imposing this condition on any subsequent users.

*/

public class Apriori extends Observable {
```

```
public static void main(String[] args) throws Exception {

Apriori ap = new Apriori(args);

}

/** the list of current itemsets */

private List<int[]> itemsets ;

/** the name of the transcation file */

private String transaFile;

/** number of different items in the dataset */

private int numItems;

/** total number of transactions in transaFile */

private int numTransactions;

/** minimum support for a frequent itemset in percentage, e.g. 0.8 */

private double minSup;

/** by default, Apriori is used with the command line interface */

private boolean usedAsLibrary = false;

/** This is the main interface to use this class as a library */

public Apriori(String[] args, Observer ob) throws Exception

{

usedAsLibrary = true;

configure(args);

this.addObserver(ob);

go();

}

/** generates the apriori itemsets from a file
```

```
* @param args configuration parameters: args[0] is a filename, args[1] the min support (e.g. 0.8 for

80%)

*/

public Apriori(String[] args) throws Exception

{

configure(args);

go();

}

/** starts the algorithm after configuration */

private void go() throws Exception {

//start timer

long start = System.currentTimeMillis();

// first we generate the candidates of size 1

createItemsetsOfSize1();

int itemsetNumber=1; //the current itemset being looked at

int nbFrequentSets=0;

while (itemsets.size()>0)

{

calculateFrequentItemsets();

if(itemsets.size()!=0)

{

nbFrequentSets+=itemsets.size();
log("Found "+itemsets.size()+" frequent itemsets of size " + itemsetNumber + " (with support

"+(minSup*100)+"%)");;

createNewItemsetsFromPreviousOnes();

} itemsetNumber++; }

//display the execution time

long end = System.currentTimeMillis();

log("Execution time is: "+((double)(end-start)/1000) + " seconds.");

log("Found "+nbFrequentSets+ " frequents sets for support "+(minSup*100)+"% (absolute

"+Math.round(numTransactions*minSup)+")");

log("Done");

}

/** triggers actions if a frequent item set has been found */

private void foundFrequentItemSet(int[] itemset, int support) {

if (usedAsLibrary) {

this.setChanged();

notifyObservers(itemset);

}

else {System.out.println(Arrays.toString(itemset) + " ("+ ((support / (double) numTransactions))+"

"+support+")");}

}

/** outputs a message in Sys.err if not used as library */

private void log(String message) {

if (!usedAsLibrary) {

System.err.println(message);
```

```
}

}

/** computes numItems, numTransactions, and sets minSup */

private void configure(String[] args) throws Exception

{

// setting transafile

if (args.length!=0) transaFile = args[0];

else transaFile = "chess.dimport java.io.*;

import java.util.*;

/** The class encapsulates an implementation of the Apriori algorithm

* to compute frequent itemsets.

*

* Datasets contains integers (>=0) separated by spaces, one transaction by line, e.g.

* 1 2 3

* 0 9

* 1 9

*

* Usage with the command line :

* $ java mining.Apriori fileName support

* $ java mining.Apriori /tmp/data.dat 0.8

* $ java mining.Apriori /tmp/data.dat 0.8 > frequent-itemsets.txt

*

* Usage as library: see {@link ExampleOfClientCodeOfApriori}
```

```
public class Apriori extends Observable {

public static void main(String[] args) throws Exception {

Apriori ap = new Apriori(args);

}

/** the list of current itemsets */

private List<int[]> itemsets ;

/** the name of the transcation file */

private String transaFile;

/** number of different items in the dataset */

private int numItems;

/** total number of transactions in transaFile */

private int numTransactions;

/** minimum support for a frequent itemset in percentage, e.g. 0.8 */

private double minSup;

/** by default, Apriori is used with the command line interface */

private boolean usedAsLibrary = false;

/** This is the main interface to use this class as a library */

public Apriori(String[] args, Observer ob) throws Exception

{

usedAsLibrary = true;
```

```
configure(args);

this.addObserver(ob);

go();

}

/** generates the apriori itemsets from a file

*

* @param args configuration parameters: args[0] is a filename, args[1] the min support (e.g. 0.8 for

80%)

*/

public Apriori(String[] args) throws Exception

{

configure(args);

go();

}

/** starts the algorithm after configuration */

private void go() throws Exception {

//start timer

long start = System.currentTimeMillis();

// first we generate the candidates of size 1

createItemsetsOfSize1();

int itemsetNumber=1; //the current itemset being looked at

int nbFrequentSets=0;

while (itemsets.size()>0)

{
```

```
calculateFrequentItemsets();

if(itemsets.size()!=0)

{

nbFrequentSets+=itemsets.size();

log("Found "+itemsets.size()+" frequent itemsets of size " + itemsetNumber + " (with support

"+(minSup*100)+"%)");;

createNewItemsetsFromPreviousOnes();

} itemsetNumber++; }

//display the execution time

long end = System.currentTimeMillis();

log("Execution time is: "+((double)(end-start)/1000) + " seconds.");

log("Found "+nbFrequentSets+ " frequents sets for support "+(minSup*100)+"% (absolute

"+Math.round(numTransactions*minSup)+")");

log("Done");

}

/** triggers actions if a frequent item set has been found */

private void foundFrequentItemSet(int[] itemset, int support) {

if (usedAsLibrary) {

this.setChanged();

notifyObservers(itemset);

}

else {System.out.println(Arrays.toString(itemset) + " ("+ ((support / (double) numTransactions))+"

"+support+")");}

}
```

- **TEST CASES**

```
 [0]  (0.6666666666666666 6)

[1]  (0.7777777777777778 7)

[2]  (0.6666666666666666 6)

[3]  (0.2222222222222222 2)

[4]  (0.2222222222222222 2)

[0, 4]  (0.2222222222222222 2)

[0, 1]  (0.4444444444444444 4)

[0, 2]  (0.4444444444444444 4)

[1, 2]  (0.4444444444444444 4)

[1, 3]  (0.2222222222222222 2)

[1, 4]  (0.2222222222222222 2)

[0, 1, 4]  (0.2222222222222222 2)

[0, 1, 2]  (0.2222222222222222 2)
```

- **PROJECT SUMMARY:**

  We found the itemsets and strong association rules. In this we developed an implemention using java for Apriori algorithm to find all frequent item sets. And atlast we generated all strong association rules from the frequent itemsets.