1. INTRODUCTION

1.1 OVERVIEW

This project is to propose a credit card fraud detection system using genetic algorithm. Genetic algorithms are evolutionary algorithms which aim at obtaining better solutions as time progresses. When a card is copied or stolen or lost and captured by fraudsters it is usually used until its available limit is depleted. Thus, rather than the number of correctly classified transactions, a solution which minimizes the total available limit on cards subject to fraud is more prominent. It aims in minimizing the false alerts using genetic algorithm where a set of interval valued parameters are optimized

1.2 PURPOSE

credit card fraud detection system is to identify suspicious events and report them to an analyst while letting normal transactions be automatically processed. For years, financial institutions have been entrusting this task to rule-based systems that employ rule sets written by experts.

2. LITERATURE SURVEY

Since the fraud detection problem has mostly been defined as a classification problem, in addition to some statistical approaches many data mining algorithms have been proposed to solve it. Among these, decision trees and artificial neural networks are the most popular ones. The study of Bolton and Hand provides a good summary of literature on fraud detection problems. However, when the problem is approached as a classification problem with variable misclassification costs as discussed above, the classical data mining algorithms are not directly applicable; either some modifications should be made on them or new algorithms developed specifically for this purpose are needed. An alternative approach could be trying to make use of general purpose meta heuristic approaches like genetic algorithms.

2.1 EXISTING SOLUTION

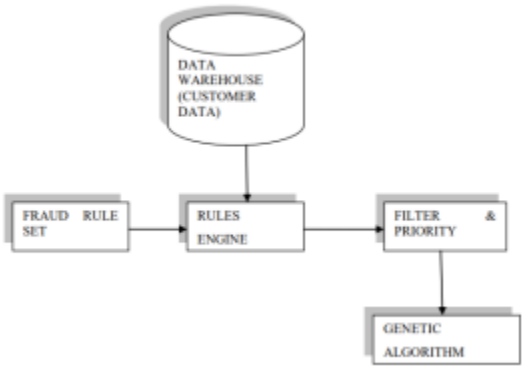
The Traditional detection method mainly depends on database system and the education of customers, which usually are delayed, inaccurate and not in-time. After that methods based on discriminate analysis and regression analysis are widely used which can detect fraud by credit rate for cardholders and credit card transaction. For a large amount of data it is not efficient.

2.2 PROPOSED SOLUTION

The proposed system overcomes the above mentioned issue in an efficient way. Using genetic algorithm the fraud is detected and the false alert is minimized and it produces an optimized result. The fraud is detected based on the customers behavior. A new classification problem which has a variable misclassification cost is introduced. Here the genetic algorithms is made where a set of interval valued parameters are optimized.

3. THEORITICAL ANALYSIS 3.1 BLOCK DIAGRAM

3.1 BLOCK DIAGRAM



3.2 HARDWARE/SOFTWARE REQUIREMENTS:

HARDWARE REQUIREMENTS

• Processor type: Pentium III-compatible processor or faster.
• Processor speed: Minimum: 1.0 GHz, Recommended: 2.0 GHz or faster
• RAM: 512 MB or more
• HARD DISK: 20GB or more
• Monitor: VGA or higher resolution 800x600 or higher resolution
• Pointing device: Microsoft Mouse or compatible pointing device

• CD-ROM: Actual requirements will vary based on system configuration and the applications and features chosen to install.

SOFTWARE REQUIREMENTS

• Application software Framework: Java
• Back End: SQL Server
• Operating System: Windows XP Professional or more

4. EXPERIMENTAL INVESTIGATIONS

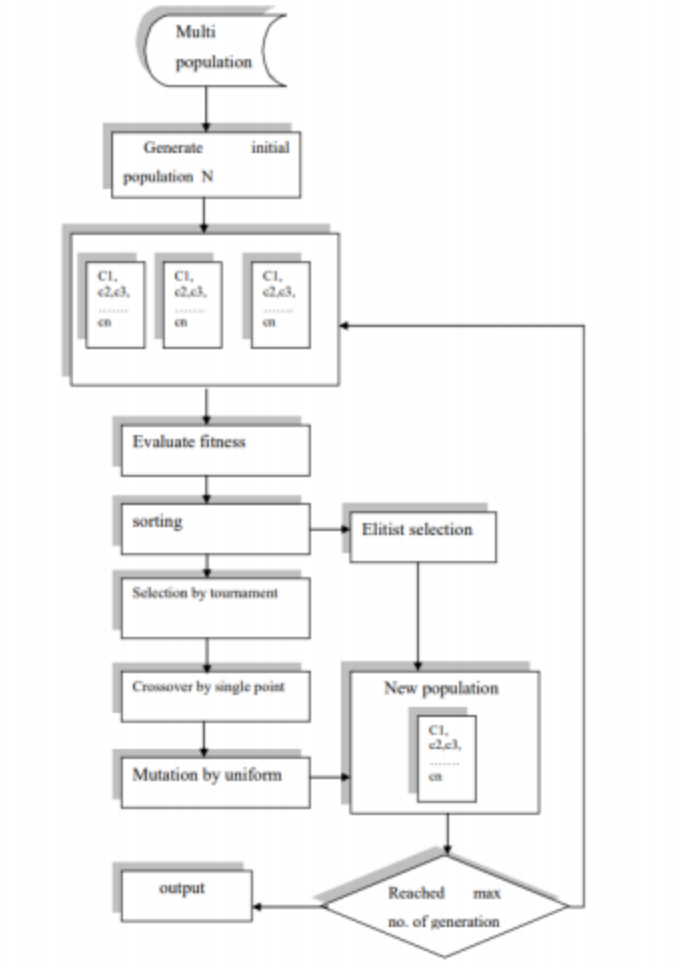
In this experimental study, we discovered several shortcomings of existing methods. We found that the approaches designed specifically to tackle the imbalance problem are not adequately effective. While these approaches improve sensitivity, this improvement leads to an increase in the number of false alarm and thus also to a drop in accuracy and AUPRC. Practically speaking, this can be costly for the financial institution just the same way a fraud event costs. Even a minimal deterioration of accuracy, say 1% hides a large misclassification rate of the majority group. The problem is summarized as follows: using imbalanced classification approaches, the number of false alarms generated is higher than the number of frauds that are detected. The results of this experimental study greatly motivated us to explore the other methods that focus on detecting the hidden patterns of fraud, with minimum misclassification.

5. VALIDATION TESTING

Validation testing is where the requirements established as part of the software requirements analysis are validated against the software that has been constructed. It provides final assurance that the software meets all functional, behavioral and performance requirements. A deviation from the specification is uncovered and corrected. Each input field was tested with the validation rules specified integrity.

6. UNIT TESTING

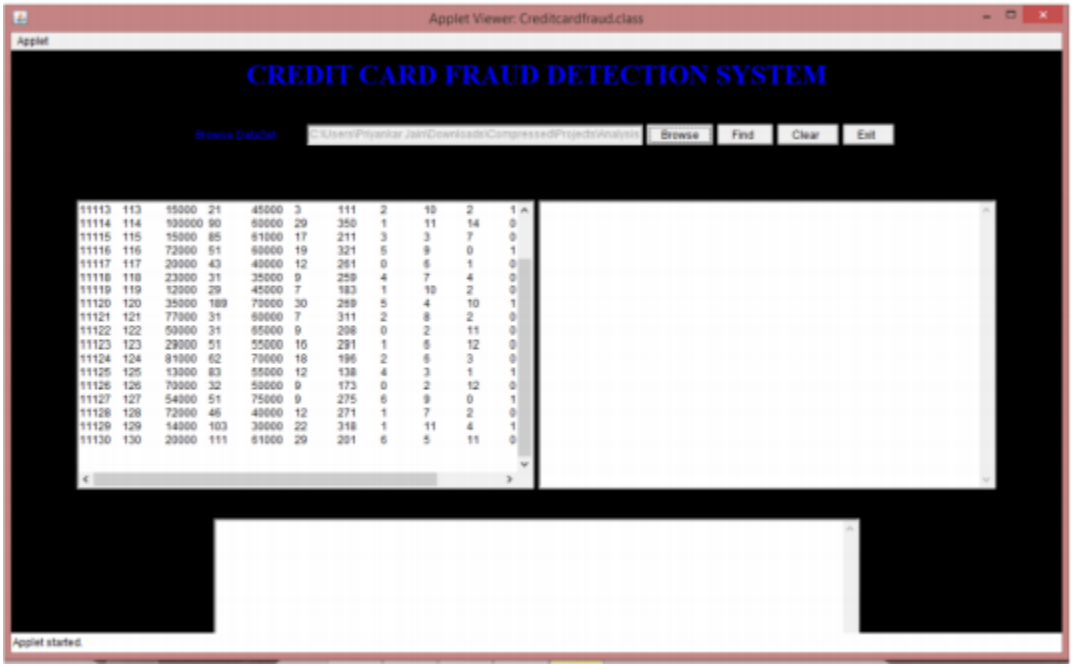
Developers write unit tests to check their own code. Unit testing differs from integration testing, which confirms that components work well together, and acceptance testing, which confirms that an application does what the customer expects it to do. Unit tests are so named because they test a single unit of code. Unit testing focuses verification effort on the smallest unit of software design. Each of the modules in this project was verified individually for errors.

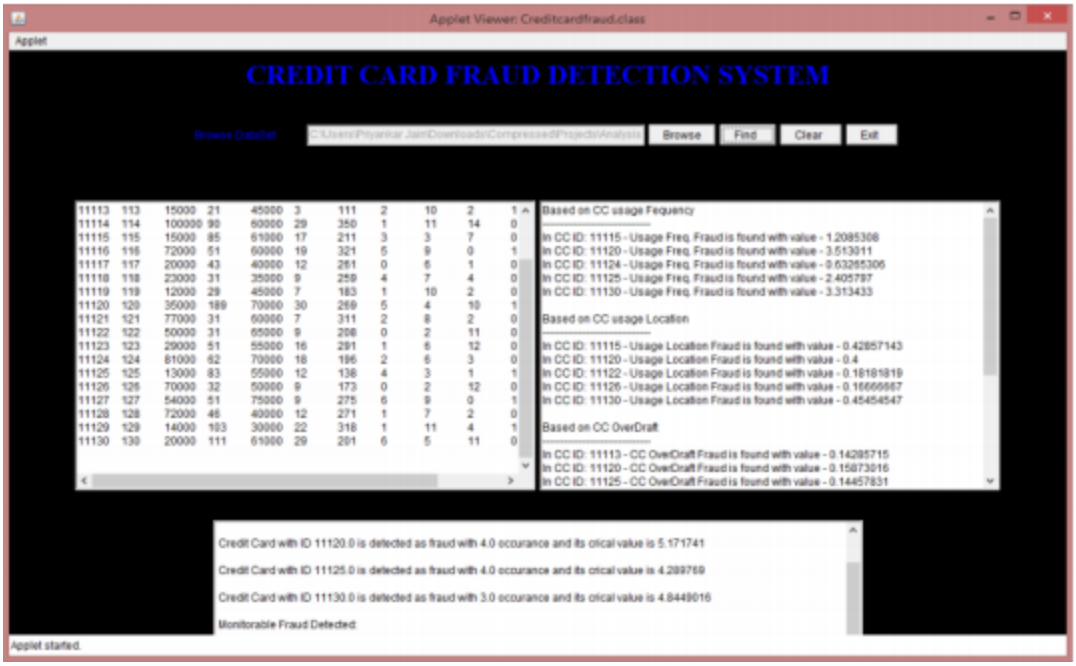


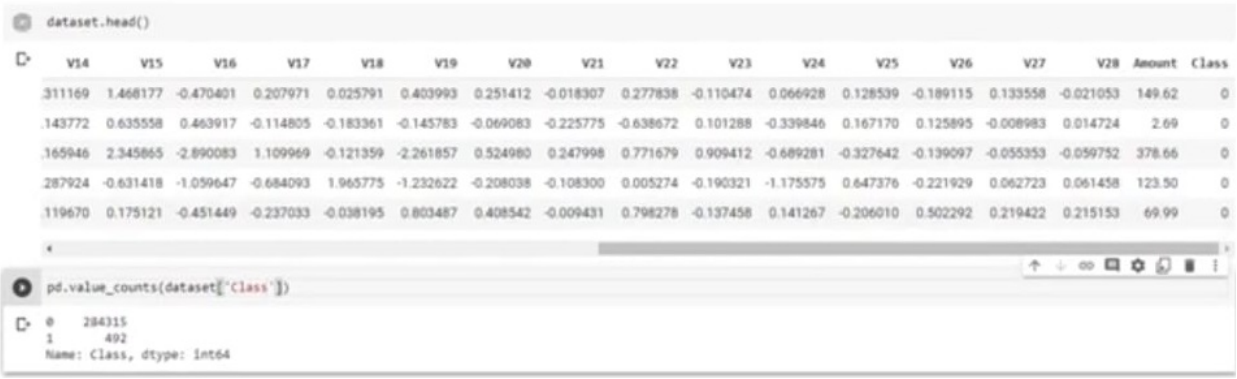
6. RESULT

The reason behind using a small dataset is that the computation cost for large datasets is high; a high-performance system is required to conduct experiments on larger dataset. The system we used for our experiments was rather a simple system and hence not able to handle high computation costs. Besides, multiple parameters affect the accuracy of this model, such as the threshold and the size of the detector set. We found a set of 2000 detectors; yet a bigger set is needed to achieve better results that would increase training time and computation cost significantly


Applet Viewer: Creditcardfraud.class
Applet
CREDIT CARD FRAUD DETECTION SYSTEM
Browse DataSet
Browse
Find
Clear
Exit
Applet started.


Applet Viewer: Creditcardfraud.class
Applet
CREDIT CARD FRAUD DETECTION SYSTEM
Browse DataSet
Browse
Find
Clear
Exit
11113 113 15000 21 45000 3 111 2 10 2 1
11114 114 100000 90 60000 29 350 1 11 14 0
11115 115 15000 85 61000 17 211 3 3 7 0
11116 116 72000 51 60000 19 321 5 9 0 1
11117 117 20000 43 40000 12 261 0 5 1 0
11118 118 23000 31 35000 9 259 4 7 4 0
11119 119 12000 29 45000 7 183 1 10 2 0
11120 120 35000 189 70000 30 269 5 4 10 1
11121 121 77000 31 60000 7 311 2 8 2 0
11122 122 59000 31 65000 9 208 0 2 11 0
11123 123 29000 51 55000 16 291 1 5 12 0
11124 124 81000 62 70000 18 196 2 6 3 0
11125 125 13000 83 55000 12 138 4 3 1 1
11126 126 70000 32 50000 9 173 0 2 12 0
11127 127 54000 51 75000 9 275 6 9 0 1
11128 128 72000 46 40000 12 271 1 7 2 0
11129 129 14000 103 30000 22 318 1 11 4 1
11130 130 20000 111 61000 29 201 6 5 11 0
Applet started.





7. ADVANTAGES AND DISADVANTAGES

- Non- availability of real data set
- Unbalanced data set
- Size of data set
- Determining the appropriate evaluation parameters

8. APPLICATIONS

- Applications for Issuing Banks • Applications for Merchants and Retailers
- Establishing these relationships between identities also helps build a customer's reputation score and is the basis to approve or disapproved transactions, according to the company

. • A mong the online activities and other details the platform checks are user purchases, transactions and orders; email, billing, shipping and IP address; payment methods; custom data and technology devices used. It also gathers information and analyzes highly-detailed behavioral patterns such as browsing patterns, keyboard preferences and screen tilt.

9. CONCLUSION

This method proves accurate in deducting fraudulent transaction and minimizing the number of false alerts. Genetic algorithm is a novel one in this literature in terms of application domain. If this algorithm is applied into bank credit card fraud detection system, the probability of fraud transactions can be predicted soon after credit card transactions. And a series of anti-fraud strategies can be adopted to prevent banks from great losses and reduce risks. The objective of the study was taken differently than the typical classification problems in that we had a variable misclassification cost. As the standard data mining algorithms does not fit well with this situation, we decided to use multi population genetic algorithm to obtain an optimized parameter

10.FUTURE SCOPE

The findings obtained here may not be generalized to the global fraud detection problem. As future work, some effective algorithm which can perform well for the classification problem with variable misclassification costs could be developed.

11.BIBILOGRAPHY

Papers:

[1] M. Hamdi Ozcelik, Ekrem Duman, Mine Isik, Tugba Cevik, improving a credit card fraud detection system using genetic algorithm, International conference on Networking and information technology 2010.

[2] Wen-Fang YU, Na Wang, Research on Credit Card Fraud Detection Model Based on Distance Sum, IEEE International Joint Conference on Artificial Intelligence 2009.

[3] Clifton phua, vincent lee1, kate smith & ross gayler, A Comprehensive Survey of Data Miningbased Fraud Detection Research,2005.

[4] Elio Lozano, Edgar Acuna, Parallel algorithms for distance-based and density-based outliers,2006.

[5] Credit card fraud detection using hidden markov model – Abinav Srivastava, Amlan

Kundu, Shamik Sural, Arun K.majumdar

Websites:

[1]http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html
[2] http://www.kxcad.net/cae_MATLAB/toolbox/gads/f6691.html
[3]http://java.sun.com/developer/onlineTraining/Programming/BasicJava1/front.html
[4]http://www.easywayserver.com/blog/user-login-in-jsp/
[5]http://www.faqs.org/patents/app/20100094765

Textbooks:

[1]Pressman, Roger S. Software engineering: a practitioner's approach / Roger S. Pressman.—5th ed.p. cm (McGraw-Hill series in computer science).
[2] E. Balagurasamy, Programming with java, Tata McGraw-Hill Publication.
[3] Ali Bahrami,Oject Oriented system Development, Tata McGraw-Hill Publication
. [4] Jiwei Han et,al., "Data Mining :Concepts and Techniques", MorganKaufmaan Series,2000

APPENDIX

SOURCE CODE

CREDIT CARD FRAUD.JAVA

```
1  import java.applet.Applet;
2  import java.awt.Button;
3  import java.awt.Color;
4  import java.awt.FileDialog;
5  import java.awt.Font;
6  import java.awt.Frame;
7  import java.awt.Graphics;
8  import java.awt.Label;
9  import java.awt.TextArea;
10 import java.awt.TextField;
```

```
11 import java.awt.event.ActionEvent;
12 import java.awt.event.ActionListener;
13 import java.io.BufferedReader;
14 import java.io.DataInputStream;
15 import java.io.File;
16 import java.io.FileInputStream;
17 import java.io.InputStreamReader;
18 import java.util.Arrays;
19 public class Creditcardfraud extends Applet implements
   ActionListener {
20 TextField brows,dname,dpath,key;
21 TextArea db,result,con;
22 Button browse,find,exit,clear;
23 int done;
24 Label browses,concl;
25 String us;
26 String strline = null;
27 String[] temp;
28 String[][] data = new String[50][50];
29 public void init(){
30 setBackground(Color.cyan);
31 setForeground(Color.magenta);
32 Label head=new Label(" CREDIT
33 CARD FRAUD DETECTION SYSTEM
34 ",Label.CENTER);
35 Font font = new Font("Serif", Font.ITALIC, 30);
36 head.setFont(font);
37 Label dataset=new Label(" DATASET SELECTED
38 ",Label.CENTER);
39 Label res=new Label(" FRAUD DETECTED
40 ",Label.CENTER);
41 browses = new Label(" Browse DataSet: ", Label.LEFT);
42 concl = new Label(" FRAUD TRANSACTIONS
43 ", Label.LEFT);
44 brows = new TextField(50);
45 db = new TextArea(20,70);
```

```
result = new TextArea(20,70);
con = new TextArea(10,100);
browse = new Button(" Browse ");
find = new Button(" Find ");
exit = new Button(" Exit ");
clear = new Button(" Clear ");
brows.disable();
resize(1200,700);
Label l1 =new Label("
");
Label l2 =new Label("
");
Label l3 =new Label(" ");
Label l4 =new Label("
");
add(head);
setForeground(Color.BLUE);
add(l2);
add(browses);
add(brows);
setForeground(Color.BLACK);
add(browse);
add(find);
add(clear);
add(exit);
// add(l1);
add(l4);
add(dataset);
add(l3);
add(res);
add(l3);
add(db);
add(result);
add(concl);
add(l3);
add(con);
```

```
82 // register to receive action events
83 browse.addActionListener(this);
84 find.addActionListener(this);
85 exit.addActionListener(this);
86 }
87 public void actionPerformed(ActionEvent ae) {
88 String str = ae.getActionCommand();
89 if(str.equals(" Browse ")) {
90 try
91 {
92 FileDialog fd = new FileDialog(new Frame(), "Please choose
   a file:", FileDialog.LOAD);
93 fd.show();
94 if (fd.getFile() != null) {
95 File f = new File(fd.getDirectory(), fd.getFile());
96 String path=f.getPath();
97 brows.setText(path);
98 FileInputStream fstream =new FileInputStream(path);
99 DataInputStream in =new DataInputStream(fstream);
100 BufferedReader br=new BufferedReader(new
   InputStreamReader(in));
101 int k=0;
102 for(int i=0;i<=20;i++)
103 {
104 strline=br.readLine();
105 temp =strline.split(",");
106 // System.out.println(temp[0] +" "+temp[1] +" "+temp[2] +"
   "+temp[3] +" "+temp[4]);
107 // System.out.println(" 1");
108 for(int j=0;j<=11;j++)
109 {
110 data[i]=temp;
111 }
112 }
113 for(int i=0;i<=20;i++)
114 {
```

```
115 for(int j=0;j<=11;j++)
116 {
117 db.append(data[i][j]);System.out.print(data[i][j]);
118 db.append("\t");System.out.print("\t");
119 }
120 if(i==0)
121 {
122 db.append("\n");
123 db.append("__\t__\t___\t\t__\t\t_\t_\t_\t_\t__\t_");
124 db.append("\n");
125 }
126 db.append("\n"); System.out.println(" ");
127 }
128 }
129 }
130 catch(Exception e)
131 {
132 System.out.println(e.toString());
133 }
134 }
135 else if(str.equals(" Exit ")) {
136 System.exit(0);
137 System.out.println("\n cancel ");
138 // repaint();
139 }
140 else if(str.equals(" Clear ")) {
141 db.setText(" ");
142 result.setText(" ");
143 brows.setText(" ");
144 System.out.println("\n Clear ");
145 // repaint();
146 }
147 else if(str.equals(" Find ")) {
148 float[] res = null;
149 float[][] fre = new float[6][20];
150 float[][] loc = new float[6][20];
```

```
151 float[][] od = new float[6][20];
152 float[][] bb = new float[6][20];
153 float[][] ds = new float[6][20];
154 float[][] initPop = new float[21][5];
155 float[][] curPop = new float[21][5];
156 float[][] nexPop = new float[21][5];
157 float[][] finalPop = new float[21][5];
158 float[] resValue =new float[21];
159 Detection dt=new Detection();
160 Evaluate ev= new Evaluate();
161 NextGen ng= new NextGen();
162 /* CC usage Fequency */
163 int l=0,m=0;
164 result.append("Based on CC usage Fequency \n");
165 result.append("------------------------------ \n");
166 for(int i=1;i<=20;i++)
167 {
168 res= dt.ccfreq(data[i]);
169 if(res[0]>=1)
170 {
171 fre[l][m]=Float.valueOf(data[i][0]);m++;
172 fre[l][m]=res[1];
173 result.append("In CC ID: "+data[i][0]+" - Usage Freq.
   Fraud is found with value - "+res[1]);
174 result.append("\n");
175 l++;m=0;
176 }
177 initPop[i][0]=res[1];
178 }
179 /* CC usage Location */
180 l=0;m=0;
181 result.append("\n");
182 result.append("Based on CC usage Location \n");
183 result.append("------------------------------ \n");
184 for(int i=1;i<=20;i++)
185 {
```

```
res= dt.ccloc(data[i]);
if(res[0]>=1)
{
loc[l][m]=Float.valueOf(data[i][0]);m++;
loc[l][m]=res[1];
result.append("In CC ID: "+data[i][0]+" - Usage Location
  Fraud is found with value -
"+res[1]);
result.append("\n");
l++;m=0;
}
initPop[i][1]=res[1];
}
/* CC OverDraft */
l=0;m=0;
result.append("\n");
result.append("Based on CC OverDraft \n");
result.append("------------------------------ \n");
for(int i=1;i<=20;i++)
{
res= dt.ccod(data[i]);
if(res[0]>=1)
{
od[l][m]=Float.valueOf(data[i][0]);m++;
od[l][m]=res[1];
result.append("In CC ID: "+data[i][0]+" - CC OverDraft
  Fraud is found with value -
"+res[1]);
result.append("\n");
l++;m=0;
}
initPop[i][2]=res[1];
}
/* Current Book Balance */
l=0;m=0;
result.append("\n");
```

```
result.append("Based on CC Book Balance \n");
result.append("------------------------------ \n");
for(int i=1;i<=20;i++)
{
res= dt.ccbb(data[i]);
if(res[0]>=1)
{
bb[l][m]=Float.valueOf(data[i][0]);m++;
bb[l][m]=res[1];
result.append("In CC ID: "+data[i][0]+" - CC Book Balance
  Fraud is found with value -
"+res[1]);
result.append("\n");
l++;m=0;
}
initPop[i][3]=res[1];
}
/* Average Daily Spending */
l=0;m=0;
result.append("\n");
result.append("Based on CC Average Daily Spending \n");
result.append("----------------------------------- \n");
for(int i=1;i<=20;i++)
{
res= dt.ccds(data[i]);
if(res[0]>=1)
{
ds[l][m]=Float.valueOf(data[i][0]);m++;
ds[l][m]=res[1];
result.append("In CC ID: "+data[i][0]+" - CC Daily
  Spending Fraud is found with value -
"+res[1]);
result.append("\n");
l++;m=0;
}
initPop[i][4]=res[1];
```

```
254 }
255 // float[][] finalresult = dt.organize(fre,loc,od,bb,ds);
256 for(int i=1;i<=20;i++)
257 {
258 for(int j=0;j<=4;j++)
259 {
260 System.out.print(initPop[i][j]);
261 System.out.print("\t ");
262 }
263 System.out.println(""); }
264 System.out.println("*** end of INIT Population ");
265 curPop=initPop;
266 for(int q=1;q<=20;q++)
267 {
268 nexPop=ng.getNextGen(curPop);
269 System.out.println(" \n");
270 System.out.println(" Current Popoulation - Generation -
   "+q);
271 System.out.println("_____________ \n");
272 for(int i=1;i<=20;i++)
273 {
274 for(int j=0;j<=4;j++)
275 {
276 System.out.print(nexPop[i][j]);
277 System.out.print("\t ");
278 }
279 System.out.println(" ");
280 }
281 curPop=nexPop;
282 System.out.println(" \n\n Critical Values Found after
   Limited number of Generations (sorted
283 order)");
284 resValue = dt.resValue(curPop);
285 Arrays.sort(resValue);
286 for(int i=1;i<=20;i++)
287 {
```

```
System.out.println(resValue[i]);
}
}
float criti=resValue[15];
float monit=resValue[10];
float ordin=resValue[5];
[3:03 PM, 5/8/2021] ganesh p cvr clg:
  System.out.println("\n\n Critical Values of each
  transaction of given DataSet");
System.out.println("
  ------------------------------------------------------
  ");
float[][] finalresult = dt.organize(fre,loc,od,bb,ds);
System.out.println("\n\n Value of Critic, Monitor and
  Ordinary Faruds");
System.out.println("\n\n "+criti+" "+monit+" "+ordin);
System.out.println(" \n\n Fraud Detected used Genetic
  Algorithm: ");
System.out.println("-------------------------------------
  ------- ");
con.append(" Critical Fraud Detected: ");con.append("\n");
con.append("---------------------------------- ");
System.out.println("Critical Fraud Detected: ");
System.out.println("----------------------------------
  ");
for(int i=0;i<=19;i++)
{
if((finalresult[i][2])>= criti)
{
con.append("\n");
con.append(" Credit Card with ID "+finalresult[i][0]+" is
  detected as fraud with
"+finalresult[i][1]+" occurance and its crical value is "+
  finalresult[i][2]);
System.out.println(" Credit Card with ID
  "+finalresult[i][0]+" is detected as fraud with
"+finalresult[i][1]+" occurance and its crical value is "+
```

```
   finalresult[i][2]);
314 [3:04 PM, 5/8/2021] ganesh p cvr clg: System.out.println("
   ");
315 con.append("\n");
316 }
317 }
318 con.append(" \n Monitorable Fraud Detected:
   ");con.append("\n");
319 con.append("---------------------------------------- ");
320 System.out.println("Monitorable Fraud Detected: ");
321 System.out.println("------------------------------- ");
322 for(int i=0;i<=19;i++)
323 {
324 if(((finalresult[i][2])>= monit) && ((finalresult[i][2])<
   criti))
325 {
326 con.append("\n");
327 con.append(" Credit Card with ID "+finalresult[i][0]+" is
   detected as fraud with
328 "+finalresult[i][1]+" occurance and its crical value is "+
   finalresult[i][2]);
329 System.out.println("Credit Card with ID
   "+finalresult[i][0]+" is detected as fraud with
330 "+finalresult[i][1]+" occurance and its crical value is "+
   finalresult[i][2]);
331 System.out.println(" ");
332 con.append("\n");
333 }
334 }
335 con.append(" \n Ordinary Fraud Detected:
   ");con.append("\n");
336 con.append("------------------------------------ ");
337 System.out.println("Ordinary Fraud Detected: ");
338 System.out.println("------------------------------------
   - ");
339 for(int i=0;i<=19;i++)
340 {
```

```
341 if(((finalresult[i][2])>= ordin) && ((finalresult[i][2])<
  monit))
342 {
343 con.append("\n");
344 con.append(" Credit Card with ID "+finalresult[i][0]+" is
  detected as fraud with
345 "+finalresult[i][1]+" occurance and its crical value is "+
  finalresult[i][2]);
346 System.out.println("Credit Card with ID
  "+finalresult[i][0]+" is detected as fraud with
347 "+finalresult[i][1]+" occurance and its crical value is "+
  finalresult[i][2]);
348 con.append("\n");
349 }
350 }
351 repaint();
352 }
353 }
354 public void paint(Graphics g) {
355 System.out.println(done);
356 if(done==1)
357 {
358 setForeground(Color.BLUE);
359 g.drawString("SUCCESS", 10, 190);
360 String msg="The File is Encrypted Successfully";
361 g.drawString(msg, 20, 205);
362 }
363 if(done==2)
364 {
365 setForeground(Color.RED);
366 g.drawString("ERROR", 10, 190);
367 String msg="The File is not Encrypted Successfully \n";
368 g.drawString(msg, 20, 205);
369 /* if(!errmsg1.equals(null))
370 {
371 g.drawString(errmsg1, 20, 220);
```

```
372 }
373 if(!errmsg2.equals(null))
374 {
375 g.drawString(errmsg2,20, 235);
376 }*/
377 System.out.println("\n paint ");
378 }
379 }
380 }
```

CONCLUSION

This method proves accurate in deducting fraudulent transaction and minimizing the number of false alert. Genetic algorithm is a novel one in this literature in terms of application domain. If this algorithm is applied into bank credit card fraud detection system, the probability of fraud transactions can be predicted soon after credit card transactions. And a series of antifraud strategies can be adopted to prevent banks from great losses and reduce risks.

The objective of the study was taken differently than the typical classification problems in that we had a variable misclassification cost. As the standard data mining algorithms does not fit well with this situation we decided to use multi population genetic algorithm to obtain an optimized parameter.

FUTURE ENHANCEMENTS

The findings obtained here may not be generalized to the global fraud detection problem. As future work, some effective algorithm which can perform well for the classification problem with variable misclassification costs could be developed.