Analysis on Credit Card Fraud Detection Methods

ABSTRACT:
Due to the rise and rapid growth of E-Commerce, use of credit cards for online purchases has dramatically increased and it caused an explosion in the credit card fraud. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of fraud associated with it are also rising. In real life, fraudulent transactions are scattered with genuine transactions and simple pattern matching techniques are not often sufficient to detect those frauds accurately. Implementation of efficient fraud detection systems has thus become imperative for all credit card issuing banks to minimize their losses. Many modern techniques based on Artificial Intelligence, Data mining, Fuzzy logic, Machine learning, Sequence Alignment, Genetic Programming etc., has evolved in detecting various credit card fraudulent transactions. A clear understanding on all these approaches will certainly lead to an efficient credit card fraud detection system. This paper presents a survey of various techniques used in credit card fraud detection mechanisms and evaluates each methodology based on certain design criteria.

EXISTING SYSTEM

- 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.
PROBLEM RECOGNITION

- The high amount of losses due to fraud and the awareness of the relation between loss and the available limit have to be reduced.
- The fraud has to be deducted in real time and the number of false alert has to be minimized.

PROPOSED SYSTEM

- 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.

SYSTEM ARCHITECTURE

HARDWARE REQUIREMENTS
• SYSTEM : Pentium IV 2.4 GHz
• HARD DISK : 40 GB
• MONITOR : 15 VGA colour
• MOUSE : Logitech.
• RAM : 256 MB
• KEYBOARD : 110 keys enhanced.

SOFTWARE REQUIREMENTS

• Operating system : Windows XP Professional
• Front End : JAVA
• Tool : NetBeans IDE

MODULES

❖ User GUI

❖ Critical Value Identification

❖ Fraud Detection using Genetic Algorithm

MODULES DESCRIPTION

❖ User GUI

In this module, User Interface module is developed using Applet Viewer. This module is developed to user to identify the credit card fraud using genetic algorithm technique. So the user interface must be capable of providing the user to upload the dataset and make manipulations and finally must show the user whether fraud has been detected or not. Only final output will be in applet screen. All the generation details (crossover and mutation) will be in the console screen of eclipse.

❖ Critical Value Identification
Based on CC usage Frequency

```java
float ccfreq = Float.valueOf(temp[3]) / Float.valueOf(temp[6]);
if(ccfreq > 0.2)
{
    if(Float.valueOf(temp[7]) > (5*ccfreq))
    {
        res[0] = 1;
        res[1] = (Float.valueOf(temp[7]) * ccfreq);
    }
}
if(res[0] < 1)
{
    res[1] = (float) ccfreq;
}
```

Ccfreq = Total number card used (CU) / CC age

If ccfreq is less than 0.2, it means this property is not applicable for fraud and critical value = ccfreq

Otherwise, it check for condition of fraud (i.e.) =

Fraud condition = number of time Card used Today (CUT) > (5 * ccfreq)

If true, there may chance for fraud using this property and its critical value is CUT * ccfreq
If false, no fraud occurrence and critical value = ccfreq

Based on CC usage Location

```java
int loc = Integer.valueOf(temp[8]);
if((loc <= 5) && (Integer.valueOf(temp[9]) > (2 * loc)))
{
    res[0] = 1;
    res[1] = (Float.valueOf(loc) / Float.valueOf(temp[9]));
}
if(res[0] < 1)
{
    res[1] = (float) 0.01;
}
```

Number of locations CC used so far (loc) obtained from dataset(loc)
If loc is less than 5, it means this property is not applicable for fraud and critical value = 0.01

Otherwise, it check for condition of fraud (i.e) =

Fraud condition = number of locations Card used Today (CUT) > (5 * loc)

If true, there may chance for fraud using this property and its critical value is loc/CUT
If flase, no fraud occurance and critical value = 0.01

**Based on CC OverDraft**

```java
float od = Float.valueOf(temp[5]) / Float.valueOf(temp[3]);
if (od <= 0.2) {
    if (Float.valueOf(temp[10]) >= 1) {
        res[0] = 1;
        res[1] = (Float.valueOf(temp[10]) * od);
    }
}
if (res[0] < 1) {
    res[1] = (float) od;
}
```

Number of times CC overdraft w.r.t CU occurred so far (od) can be found as,

**Od w.r.t CU = OD/CU**

If Od w.r.t CU is less than 0.02, it means this property is not applicable for fraud and critical value = Od w.r.t CU

Otherwise, it check for condition of fraud (i.e) =

Fraud condition = check whether overdraft condition occurred today from (ODT dataset)

If true, there may chance for fraud using this property and its critical value is ODT * Od w.r.t CU
If flase, no fraud occurance and critical value = Od w.r.t CU

**Based on CC Book Balance**
float bb = Float.valueOf(temp[2])/Float.valueOf(temp[4]);
if(bb<=0.25)
{
    res[0]=1;
    res[1]=(Float.valueOf(2)*bb);
}
if(res[0]<1)
{
    res[1]=(float)bb;
}

Standard Book balance can be found as,

\[ Bb = \frac{\text{current BB}}{\text{Avg. BB}} \]

If \( bb \) is less or equals than 0.25, it means this property is not applicable for fraud and critical value = \( bb \)

Otherwise, it check for condition of fraud (i.e) =

If true, there may chance for fraud using this property and its critical value is currBB * \( bb \)
If false, no fraud occurrence and critical value = \( bb \)

**Based on CC Average Daily Spending**

float mon= Float.valueOf(temp[6])/30;
float bal= 100000 - Float.valueOf(temp[4]);
float tot = mon*bal;
float ds =tot/Float.valueOf(temp[6]);
if((10*ds)<Float.valueOf(temp[11]))
{
    res[0]=1;
    if(Float.valueOf(temp[11])>0)
        res[1]=(Float.valueOf(temp[11])/ (10*ds));
    else
        res[1]=(float) 0.0;
}
if(res[0]<1)
{
    res[1]=(float)0.01;
}
Fraud Detection using Genetic Algorithm

In this module the system must detect whether any fraud has been occurred in the transaction or not. It must also display the user about the result. It is calculated based on following:

Age of CC in months can be calculated using CCage (from dataset) by,

\[ \text{Age of cc by month} = \frac{\text{CCage}}{30} \]

Total money being spent from the available limit (1 lakh _ 100000)

\[ \text{Bal} = 100000 - \text{avg BB} \]

So, total money spent can be found as,

\[ \text{Tot} = \text{Age of cc by month} \times \text{Bal} \]

Total money spent on each month can be calculated as,

\[ \text{Ds} = \text{tot} \times \text{Age of cc by month} \]

it check for condition of fraud (i.e) =

\[ \text{Fraud condition} = 10 \times \text{ds} \]

is amount spent today (AmtT in dataset)

If true, there may chance for fraud using this property and its critical value is

\[ \text{AmtT}/(10*\text{ds}) \]

If false, no fraud occurrence and critical value 0.01

REFERENCE: