Database Implementation and Testing of Dynamic Credit Card Fraud Detection System

Anita Jog
PG Student,
MIT College of engineering,
Kothrud, Pune.

Anjali Chandavale, PhD
Professor,
Department of Information Technology,
MIT College of Engineering, Pune

ABSTRACT
Credit card frauds are increasing with the increase in use of plastic money. These frauds include the transactions done either by stealing the physical card or using card data such as card number, expiry date and pin number. There is a need to recognize customer spending pattern and apply validations for incoming transaction. Suspicious transactions can go under rigorous security checks. This paper describes the database implementation of credit card fraud detection system which is adaptive to concept drift environment. The system is designed using PL-SQL stored procedures and JAVA. The validation procedure and testing results are included in this paper.

General Terms
Pattern Recognition, Security, credit card, fraud detection, stored procedure.

Keywords
Concept drift, self learning, credit card fraud detection.

1. INTRODUCTION
There is tremendous growth in the use of Credit cards. People are encouraged to use plastic money to control the corruption. Apart from the corruption control issue, credit card is gaining popularity due to online shopping trend. Retailers, merchants are offering discounts on online shopping. Customers prefer Online shopping since it helps explore many items with few clicks. Customers also compare the amount charged by different vendors for the same thing.

Personal details are exposed over the network during online transactions. It results in loss of heavy monetary value worldwide every year. As per cybercrime report [13], the ratio of transaction volume ($28 Trillion) vs fraud transactions ($16 billion) is 0.06%. I.e. 19% increase in fraud transactions while the customer base grew by 15%. So it is utmost priority for electronic transactions processing Companies to maintain customer trust and protect their business by smartly detecting frauds. The important aspect to prevent the credit card fraud is to analyze the customer spending pattern thoroughly and apply validation rules to categorize the transaction to be either fraud or genuine. The paper includes following sections. Section 2 describes the related work. Section 3 detailed out the proposed work. In section 4 shows results, Section 5 provides end conclusion, and Section 6 includes referred papers and sites.

2. RELATED WORK
There are several data mining techniques suggested for fraud detection [8][6][1][11]. Artificial Intelligence, Neural networks, genetic programming, Support Vector machine, Decision tree. etc. [2][3][7][10]. Véronique Van Vlasselaer and Cristián Bravo [5] has suggested the approach which combines inherent attributes derived from the characteristics of incoming transactions and the customer spending history using the primary characteristics such as Recency–Frequency–Monetary. Also, the network of credit card holders and merchants is taken into account to validate their relationship by calculating time-dependent suspiciousness score for each network object. Intrinsic feature extraction is implemented using supervised learning by exploring spending patterns. Serol Bulkan and Yusuf Sahin describe a system that makes use of cost sensitive decision tree approach. The approach minimizes total misclassification costs but also identifies splitting attribute at non-terminal nodes [9]. Author has compared this approach against traditional classification models on real life data. Yiğit Kültür[15] has focused on analyzing the cardholder spending behavior and proposes a novel cardholder behavior model for detecting credit card fraud. The model is named Cardholder Behavior Model (CBM). He has used sensitivity, specificity, false positive rate, precision, accuracy to evaluate the customer behavior model.

3. PROPOSED WORK
The proposed and implemented solution is built on above mentioned fundamental solution for fraud detection. It is based on the layered architecture. The different layers used are 1) data, 2) utility, 3) manager and 4) controller. Data layer stores following data types 1) Historical, 2) Transactional. The validation model contains rule set to validate each customer. Data layer also includes stored procedures that are used for validation purpose. Utility layer contains common utilities supporting other layers. Manager layer is used for executing each task independently while controller wraps all the small tasks that need to be done for every user action. This technique (Concept Drift Adaptation) is implemented by periodically updating the model using scheduler. Following section explains flow of credit card transaction followed by implementation details of the proposed system.

3.1 Overview of credit card processing
There are 4 basic steps involved in credit card processing. First one is authorization. When customer swipes the card, his credentials are sent to the bank with which the card machine is registered and then subsequently forwarded to the card issuing bank. Card issuing banks authenticates the request and depending upon its response, transaction can either proceed or denied for sale. Second step is batching where merchant groups all day's transactions and submits to the bank for payment processing. Third step is clearing where group of transactions received are segregated and sent to appropriate bank through card network. Card issuing bank deducts the interchange fee and sends remaining amount through the network. Last step is

Funding in which merchant's bank subtracts appropriate charges and transfers the money to merchant's account.

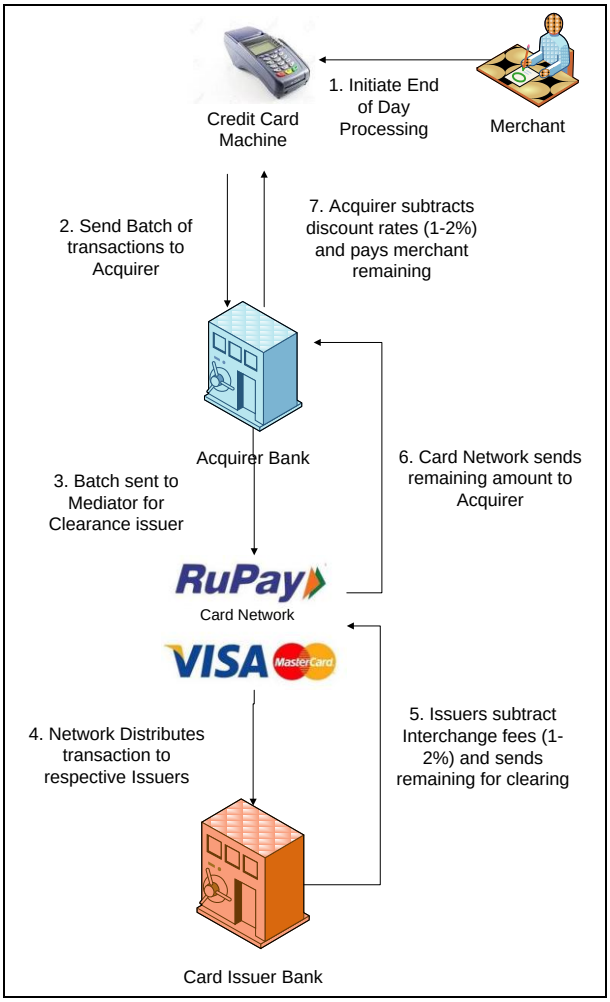


Figure 1: Overview of credit card processing

3.2 Proposed System

In a nutshell the component Builder uses historical transactional data to build the model. This model is a set of attributes for identifying fraudulent transaction. The Online fraud detected uses this model to detect suspicious characteristics in each incoming transaction. Based on the decision from online fraud detector, another component called transaction processor aborts or proceeds with the transaction for further processing. Next component called offline calibrator is a scheduler which runs periodically. Offline calibrator is responsible for rebuilding the model using transactions and fraud detections after previous run of the scheduler. This model is updated weekly to cater newly added records. The application consists of modules such as Login, Account Statement Viewer, PDF and Excel Download to actual transaction execution covering various validations. Validations are performed at 2 intervals, 1) During actual transaction and 2) periodically heuristic checks are applied across entire customers set to recognize suspicious Pattern of transaction.

Figure 2 shows the architecture diagram of the proposed system. The system is mainly divided into four layers Controller layer, Manager Layer, Utility layer and data layer.

Controller Layer wraps all tasks that need to be performed for every user action. It sends the request to appropriate manager to execute each action sequentially. It takes help from alert utility depending upon the response received from manager layer. Each controller class represents one user action. It determines necessary steps that need to be executed to complete given task.

Manager Layer is responsible for executing below mentioned action with the help of database tables and stored procedures. It sends response code to the controller. Authentication Manager is the first manager that gets called from controller. Authentication manager is used to verify the userId, password. The authentication manager returns error message in case of invalid user or password. It returns the customer details if the user id, password is valid. After passing all the preliminary checks, controller calls fraud detection manager to detect any suspicious behavior of transaction. It calls stored procedures written in PL-SQL. At the last, controller calls Transaction manager to process the transaction with respect to debit or credit etc.

Utility Layer is responsible of reporting, notifications, alerts triggered by manager layer.

Database Layer includes transactional data, historical data, and set of stored procedures. Transactions that are marked suspicious by the algorithm are kept in separate table fraud_log_tbl. Following section tells about database architecture.

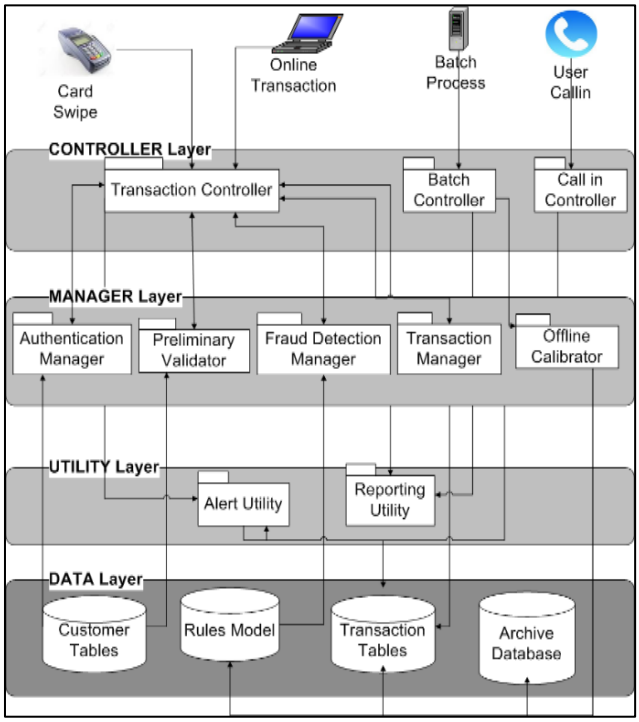


Figure 2: Proposed system architecture

3.3 Database Architecture

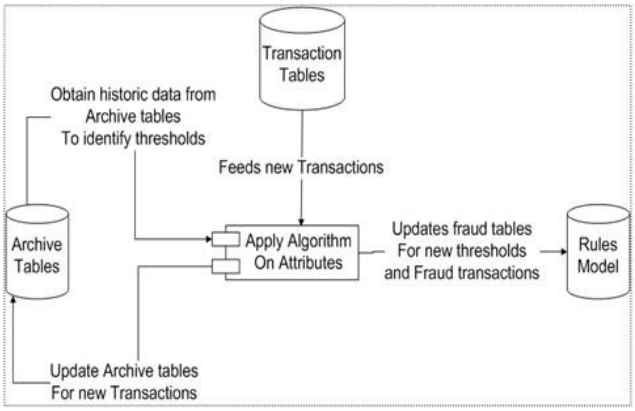


Figure 3: Database architecture

Figure 3 gives the quick overview of database architecture. The historical transactional data is analyzed and derived known patterns are stored in database tables. These act as validation rules to identify the suspicious transaction. These validation rules are explained below in stored procedure section. The patterns are nothing but characteristics obtained from customers' historical transactions to identify the spending pattern. These characteristics are stored in various tables as mentioned below in Figure 4.

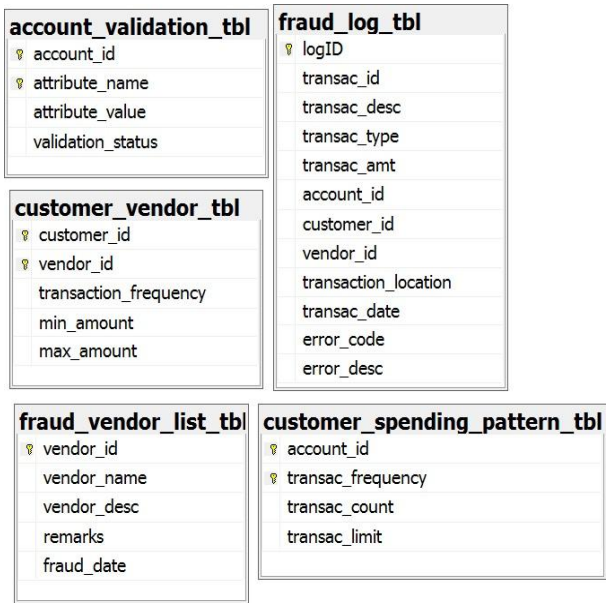


Figure 4: Important tables involved in credit card fraud detection

Customer_Vendor_tbl stores RFM attributes (Recency, Frequency, and Monetary) value for each customer-vendor pair. The incoming transaction is validated against attributes to check if it is within the threshold value. For example, if it is quarterly, then last transaction date and current date for same vendor is compared. If it is found that the difference between two transactions is less than a quarter, then after adding 20% variation, transaction is put on HOLD. Customer feedback is needed to decide how to process suspicious transactions. Customer care will work with customer to mark it genuine or fraud. *Account_validation_tbl* stores various attribute level validations. I.e. Min/Max amount, Location etc. e.g. For location it stores delimited locations list. If a transaction comes from a new city then it goes in HOLD. The design allows customized attributes by Account. *Fraud_log_tbl* logs all Blocked transactions. *customer_spending_pattern_tbl* is used to store he customer spending behavior such as how many transactions customer performs in a day, how much amount spent daily, how much amount spent weekly, number of transactions executed in a week, weekly amount limit, monthly transactions, monthly limit.

Below are main stored procedures which do the job of identifying the fraudulent transaction. These stored procedures are divided into three modules as shown in table below.

Details of each stored procedure are explained below.

3.3.1 perform_HeuristicSearchforVendor:

This stored procedure performs heuristic search across all the accounts to find the pattern of similar transactions within same time period, amount, and vendor. E.g. online hackers are known to do small and similar amount transactions across millions of accounts. Customers generally are not even aware of such frauds if they do not monitor the account regularly however the hackers earn millions at the loss of credit-card issuer. In these cases the suspected transactions are put on HOLD and kept in different table. After customer confirmation, these transactions can proceed or aborted. Similarly such a vendor will also marked as Fraud i.e. restricting any further transactions. If there are more than 100 transactions within 2 hours having amount difference less than 10, then these transactions could be due to hacking. So these transactions are marked as suspicious

Table 1: Overview of stored procedures

Stored Procedure Name	Function
populate_AccountValidationTbl perform_HeuristicSearchforVendor populate_CustomerSpendingPatternTbl populate_RFMAttributeTbl	This stored procedure populates the data in "account_validation_tbl" using historical transactions. This table contains attributes for each account to validate the transaction. Attributes include minimum amount spent, maximum amount spent, location where customer perform the transaction etc.
validateCustomerPattern validate_AccountAttributes validate_RFMAttributes	To apply the rules built by first module on the incoming transaction to check if it is suspicious or genuine
bulkpopulate_TestValidationTbl	Test all the conditions

3.3.2 Populate_RFMAttributeTbl:

This stored procedure populates the Recency, Frequency, and Monitory values for each Customer-Vendor pair. It populates the minimum and maximum transaction amounts, Frequency of transactions that customer deal with particular vendor. It reads the records form transaction table and group the transactions by customer Id, Vendor Id. Within the group, the count of transactions is considered. Depending upon the

count, the frequency is calculated as shown in below table; One year data is taken into account while doing this calculation.

Table 2: Customer-Vendor transaction frequency calculation

No of transactions for customer-vendor group	Frequency
Transaction Count >100	Daily
Transaction count between 45 to 100	Weekly
Transaction count between 20 to 44	Bi-weekly
Transaction count between 9 to 19	Monthly
Transaction count between 3 to 8	Quarterly
Transaction count < 3	Yearly

3.3.3 Populate_AccountValidation & SP_Populate_AccountValidationAll:

This stored procedure populates the transacted cities, minimum amount, and maximum amount values for each Account. First one is specific to an Account, whereas the later one does the same operation across ALL Accounts.

From transaction history, distinct locations are found from where customer usually does the transactions for particular account. The locations are collected and made a list of cities separated by comma. This list is stored in accountValidationTbl table along with what is the minimum amount, maximum amount.

3.3.4 Populate_CustomerSpendingPattern:

This stored procedure populates the Customer's spending pattern i.e. weekly/monthly/daily transaction count, minimum amount, and maximum amount spent against each Account. This tracks down the customer spending pattern in a specific period. For example, customer usually buys breakfast from cafeteria. Then while coming back from office, he buys tea or some snacks from another vendor. Also shops some grocery items from grocery store. So daily transaction count will be 3 and daily amount limit will be in between 60 to 200. Every week, he goes to Big Bazar for shopping, fills petrol in the car etc. So weekly count will be 2 and amount will be in the range of 3000 to 4500. Similarly the monthly pattern can be derived where the electricity bill is paid, car wash charges are paid, and parking or toll charges are paid. Thus typical customer spending pattern is derived and stored in *customer_spending_pattern_tbl*. To calculate the daily, weekly transaction count, this stored procedure first calculates the number of transactions done in every week for that account over one year. Then average is calculated.

3.3.5 validateCustomerPattern:

This stored procedure validates the incoming transaction against the data in *customer_spending_pattern_tbl* table. If customer usually spends around 200 Rs. every day in 2 transactions then suddenly there are 5 transactions happens then there are more chances that the credit card is stolen or someone else is using credit card details to buy the things. So all these suspicious transactions are put on HOLD and wait for customer confirmation. Similarly if customer goes and buys a single item but of 1000Rs, then also, it is suspicious since daily limit is 200Rs. For this when new transaction request is received, the number of transactions are calculated that are done in same day, number of transactions done in same week, number of transactions done in the same month. Along with the number of transactions, amount also is calculated for day, week and month spent by that customer. Now the amount of current transaction is added and then it is compared against daily limit stored in *customerSpendingPatternTbl.* The count is also increased by 1 to consider current transaction. This count is compared against daily transaction limit stored in *customerSpendingPatternTbl.* Similar checks are applied for weekly count, weekly amount limit, monthly count and monthly amount limit.

3.3.6 validate_AccountAttributes:

This stored procedure validates the incoming transaction against "*account_validation_tbl*" table. This attributes are common values such as minimum amount, maximum amount, location of transaction etc. If customer lives in Pune and works at Hinjewadi, his most probable location is Pune. If he travels to Mumbai once a while, his list of locations will contain Pune, Mumbai. So if there is sudden transaction from Delhi, it is marked as suspicious and put on HOLD.

3.3.7 validate_RFMAttributes:

This stored procedure validates the customer-vendor specific attributes. From historical data, *customer_vendor_tbl* is populated by "*populate_RFMAttribute*". "*customer_vendor_tbl*" has the customer-vendor relationship. This includes what is the frequency of transaction with particular vendor, what is the amount usually customer spend with the vendor. For example, the electricity bill is paid monthly and suddenly there are 3 transactions with electricity bill in same month, it is marked as suspicious. Or customer visits D-Mart every month for grocery and 3 transactions with amount greater than given threshold happens, it is marked as suspicious.

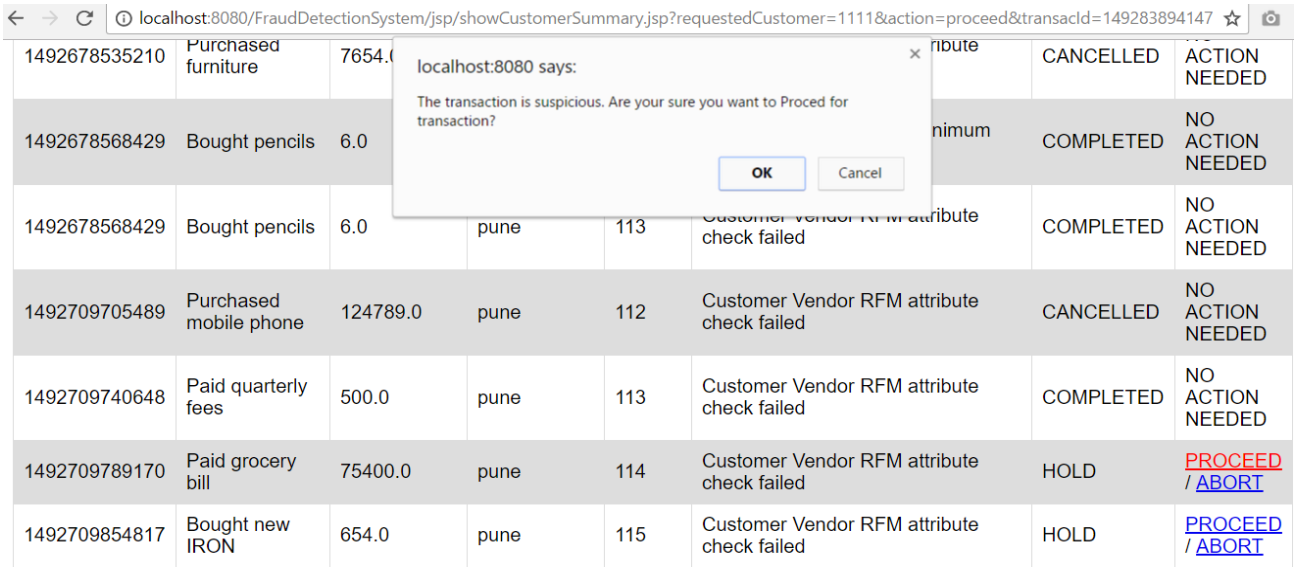
3.3.8 bulkpopulate_TestValidationTbl:

This stored is specially designed for testing various flows from backend database and for performance testing. Hundreds of transactions can be tested with this stored procedure one by one. It reads batch of transactions from transaction table and apply all the validations one by one to check if it is suspicious.

4. RESULT

The graphical user interface (GUI) is also designed to test a particular transaction to be a fraud or genuine. Validation results are displayed.The transactions that are put on HOLD can proceed or abort depending upon customer feedback. If customer confirms the transaction to be genuine then transaction processing is completed by entering in *Transactiontbl* table. Secondly the status is marked as COMPLETED in *fraud_log_tbl*. And lastly the stored procedures are executed to recalibrate the validation rules. This is to prevent marking similar transaction as suspicious in future. If customer confirms the transaction to be fraudulent, then transaction status is updated in *fraud_log_tbl* as cancelled. And no further action is taken on this.

The data is obtained from kaggle.com. Dataset includes transactions made by credit cards by European cardholders. With some modifications synthetic data is also inserted into the database to validate different scenarios. *test_transaction_tbl* table is created to collect the test results. Each column in this table represents status of each validation rule. *bulkpopulate_TestValidationTbl* is a stored procedure specially designed to test various scenarios. This also gives the performance parameters to check how much time system will take for checking thousands of transactions.



1492678535210	Purchased furniture	7654.(			ribute	CANCELLED	ACTION NEEDED
1492678568429	Bought pencils	6.0			nimum	COMPLETED	NO ACTION NEEDED
1492678568429	Bought pencils	6.0	pune	113	ribute check failed	COMPLETED	NO ACTION NEEDED
1492709705489	Purchased mobile phone	124789.0	pune	112	Customer Vendor RFM attribute check failed	CANCELLED	NO ACTION NEEDED
1492709740648	Paid quarterly fees	500.0	pune	113	Customer Vendor RFM attribute check failed	COMPLETED	NO ACTION NEEDED
1492709789170	Paid grocery bill	75400.0	pune	114	Customer Vendor RFM attribute check failed	HOLD	PROCEED / ABORT
1492709854817	Bought new IRON	654.0	pune	115	Customer Vendor RFM attribute check failed	HOLD	PROCEED / ABORT

Figure 5: screen capture showing fraud summary page to view transactions that are HOLD

Table 3: Performance result

No of transactions	Execution Time (ms)
1046	1777
1884	3872
2730	6360
3576	9663
4422	13470

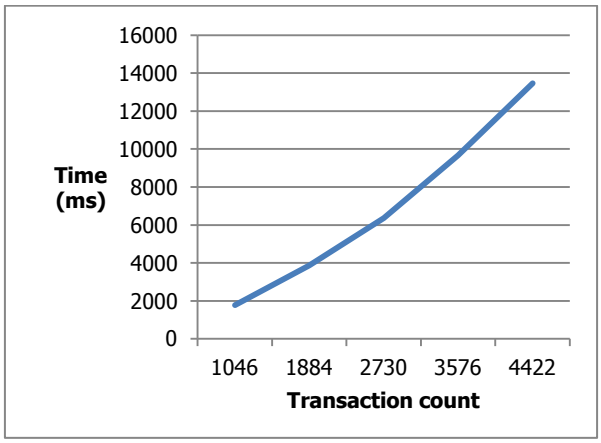


Figure 6: Performance graph

Parameters used to measure the system accuracy are sensitivity, precision, false positive rate, negative predictive value and accuracy as shown in table 2. [15]

Table 4: System evaluation result

Sensitivity	Precision	Accuracy
42.24%	10.20%	81.12%

5. CONCLUSION

Self-Learning algorithms for fraud detection in credit cards is a need of the hour as Plastic and Online transaction base is growing at a fast pace. There is also a need for periodically recalibration of Algorithm. The approach suggested above is suitable in such an environment since it recalibrates, and customizes by logical entity. It is a hybrid system having Network based extension and concept drift adaptation. The model is customized based on recent and past transactions. The suggested model is dynamic and customized with 81% accuracy in transactions filtering rate. Thus, the system is providing necessary support system to end users and credit card companies to freely use Plastic and electronic money.

6. REFERENCES

[1] Emanuel MinedaCarneiro, "Cluster Analysis and Artificial Neural Networks: A Case Study in Credit Card Fraud Detection," in 2015 IEEE International Con- ference

[2] V. Mareeswari, "Prevention of Credit Card Fraud Detection based on HSVM". 2016 IEEE International Conference On Information Communication And Em- bedded System.

[3] Carlos A. S. Assis, "A Genetic Programming Approach for Fraud Detection in Electronic Transactions" in Advances in Computing and Communication Engi- neering (ICACCE), 2015 Second International Conference

[4] Andrea Dal Pozzolo, "Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information",

[5] Véronique Van Vlasselaer, "APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions" published in Deci- sion Support Systems 2015

[6] Dhiya Al-Jumeily, "Methods and Techniques to Support the Development of Fraud Detection System", IEEE 2015

[7] Dustin Y. Harvey, "Automated Feature Design for Numeric Sequence Classifica- tion by Genetic Programming", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 19, NO. 4, AUGUST 2015

[8] Mukesh Kumar Mishra, "A Comparative Study of Chebyshev Functional Link Artificial Neural Network, Multi-layer Perceptron and Decision Tree for Credit Card Fraud Detection", 2014 13th International Conference on Information Technology

[9] Sahin Yusuf, BulkanSerol, DumanEkrem,"A Cost-Sensitive Decision Tree Approach for Fraud Detection", Expert Systems with Applications, vol.40, pp.5916-5923, 2013

[10] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang, "Credit Card Fraud Detection Using Convolutional Neural Networks", Neural Information Processing, Springer [11] Andrea Dal Pozzolo, Olivier Caelen," Learned lessons in credit card fraud detec- tion from a practitioner perspective" , Expert Systems with Applications 41,2014.

[12] How a credit card is processed https://www.creditcards.com/credit-card-news/assets/HowACreditCardIsProcessed.pdf

[13] Global Card Fraud Damages Reach $16B http://www.pymnts.com/news/2015/global-card-fraud-damages-reach-16b/

[14] Credit Card Fraud Detection https://www.kaggle.com/dalpozz/creditcardfraud

[15] Yiğit Kültür, "A Novel Cardholder Behavior Model for Detecting Credit Card Fraud", IEEE international conference on commuting and communication engineering, 2015