International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 6, Issue 4, 2019, PP 7-15
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0604002
www.arcjournals.org

On Reducing Misclassifications and Error on CART Models using Rtosynr Dataset for Improved Debit Card Fraud Detection

Nwogu E. R[1*], Nwachukwu E. O.[2], Ejiofor V. E. [3]

[1,2] *Department of Computer Science, University of Port-Harcourt, Nigeria*

[3] *Department of Computer Science Nnamdi Azikiwe University Awka, Nigeria*

Corresponding Author:** ***Nwogu E. R, *Department of Computer Science, University of Port-Harcourt, Nigeria*

Abstract: *This work describes a way of reducing the number of misclassifications in Classification and Regression Tree (CART) classifiers for debit card fraud detection using the Real-to-Synthetic-Real (RtoSynR) model; thereby increasing the classification accuracy of the classifier. RtoSynR model involves the generating of synthetic transaction data from a sample of real data to augment the available real data during the training of the model. The joining of the generated synthetic data with the available real data produces the RtoSynR dataset which is used to train and improve the classification accuracy of the classifier. The RtoSynR dataset is updated first by the customer feedback and then by the modeling and generating of newly observed transaction patterns (fraudulent and non-fraudulent patterns) to solve the problem of class imbalance in the training set and concept drift in machine learning which results in underfitting and overfitting classifiers. The detailed algorithm for RtoSynR implementation was described and implemented. Result of the tests shows an improvement on the classification accuracy of the classifiers developed with RtoSynR dataset over those developed with real dataset.*

Keywords: *CART, Debit card, Credit card, Model, RtoSynR, Synthetic*

1. INTRODUCTION

Since the application of machine learning in fraud detection systems, emphasis has been on the improvement of the classification accuracy of these systems. This has sustained interests in research on better and efficient fraud detection and prevention models. Very important is also the fact that fraud detection systems need to continuously evolve to prevent the possibility of fraudsters breaking into the system (Sonepat and Sonepat, 2011). This view is also shared by Lepoivre etal. (2016) who opines that fraudulent techniques are changing over time, and thus detection system needs to be adaptive to maintain its efficiency. Similarly, Meshram & Bhanarkar (2012) posit that fraud detection is beyond detecting the fraud, but detecting it as rapidly as possible. Over the years, credit and debit cards have continued to be the main target of financial fraudsters going by the number of users of this payment system worldwide corrobrating the increasing rate of card fraud is Bolton & Hand (2002) and thus posits that active surveillance of these systems needs to be in place. There has been such issues about the efficiency of these systems, espcially the cost effectivenes of deploying fraud detection systems. This view was shared by Jon and Sriganesh (2008) who reported that cost of transaction screening should not be higher than or close to the possible loss as a result of the fraud.

Despite the challenge of developing an efficient model and system for debit card fraud detection, some challenges have hampered research in this subject. This has been the unavailability of transaction data for fraud detection research. As posited by Zhang etal. (2015)**,** getting adequately sufficient labeled real data for the training of a classifier is often difficult in real world scenarios. The few labeled transaction data available still have the class imbalance problem (Haibo and Edwardo, 2009); making it difficult to conduct research on better and efficient fraud detection systems. Apparently as has been generally accepted in the research community, the greatest challenge posed by the unavailability of transaction data is the class imbalance problem, where records of the majority class (usually the non-fraudulent class in fraud detection systems) greatly outnumbers that of the minority class (usually the fraudulent class). This potentially leads to conditions known as overfitting and underfitting in the training algorithm, making it impossible for the model to correctly classify

incoming transactions. Supporting this is Andrea et al. (2014), who in their work reported that Machine Learning algorithms do not work well with unbalanced or overlapped class distributions.

Similarly, the problem of concept drift is another challenge in machine learning. This happens when there is a sudden change in the characteristics of data making the current model unable to classify incoming transactions. According to Webb et al. (2018) Concept drift and shift are major issues that greatly affect the accuracy and reliability of many real-world applications of machine learning. Also Tsymbal (2004) reported that concept drift complicates the task of learning a model from data and requires special approaches, different from commonly used techniques.

The use of synthetic data in building machine learning models has been proposed and supported by several authors in literature. Researchers have found out the problem of real data unavailability can be solved by using synthetically generated data. Neha et al. (2016) reported no statistically significant difference between the accuracy scores of data scientists with control data and data scientists with synthesized data. This view has also been shared by Lopez-Rojas and Axelsson (2012), who in their work reported no significant difference between synthetically generated data and real data; concluding that synthetically generated data can successfully replace real data especially in the field of fraud detection and prevention research.

In this work, we have deployed the ***RtoSynR*** algorithm on Classification and Regression Trees (CART) algorithm Breiman et al. (1984). ***RtoSynR*** is a method of generating synthetic data and combining the synthetic data with the available real data to solve the problem of unavailability of training data, class imbalance and concept drift in machine learning, thereby reducing misclassifications in decision tree algorithms.

1.1. Classification and Regression Tree Algorithm

The CART algorithm was developed by (Breiman etal., 1984) and uses the Gini index as its splitting function. CART normally constructs binary trees only, with each node having two outgoing edges. CART uses weighted average impurity to select splits (Masa and Kocza, 2006), with the obtained tree pruned by cost-complexity pruning. CART also has the ability to generate regression trees, which are trees that predict real number values instead of classes. In regression trees, CART uses splits that minimize the prediction squared error. CART has several advantages as listed in Bhumika (2017) and Jatinder and Jasmeet (2015)**.** The attribute with the least value of Gini index is the most suitable for the split operation.

In classification trees, the impurity measure defines the correctness of a split. A pure split arises when after a split, for all branches all instances at a branch belong to the same class.

Taking N as the number of training instances at the root node, while N_m is the number of training instances reaching node m. Let us also assume that N^i_m is the number of training instances belonging to class i at node m, while p^i_m is the probability of samples belonging to class i at node m; we can summarily say that N^i_m of N_m belong to class C_i; we can then solve the estimate of the probability of class i at node m according to Alpaydin (2010**)** with equation 1.

$$\hat{P}(C_i|x,m) \equiv p^i_m = (N^i_m/N_m) \quad (1)$$

Given equation 2.15 above, node m is pure if p^i_m for all i are either 0 or 1. At this point no further split is required. A number of methods have been proposed and used in impurity measurement. Quinlan (1986) proposed entropy as measure of impurity in classification learning. Consequently we have equation 2

$$I_m = Entropy(N) \equiv - p^i_m \log_2 p^i_m \quad (2)$$

Where N is the number of sample or collection, k is the number of classes in the sample. For a binary classification, we have entropy given by equation 3

$$I_m = Entropy(N) \equiv - p^1_m \log_2 p^1_m - p^2_m \log_2 p^2_m \quad (3)$$

Where p^1_m is the proportion of the positive class at node m and p^2_m the proportion of the negative class at node m. Entropy is maximum when the two classes are evenly distributed, i.e $p^1_m = p^2_m$ with the graph below showing entropy function for different values of the proportion of the positive class in binary classification learning.

Similarly, the Gini index has also been used in binary classification impurity computation. This has been applied in Classification and Regression Tree (CART) algorithm (Breiman etal., 1984). It was developed by Italian statistician Corrado Gini in 1912 (Barry and Padraic, 2013).

Gini impurity is computed by summing the probability p^i of choosing an item with label i multiplied by the probability of a mistake (1- p^i) in categorizing the item. The minimum (zero) is reached when all cases in the node fall into the same class.

$$I_G(p) = = (pi - pi^2) = pi - pi^2 = 1 - pi^2$$

Where *pi* is the probability of choosing an item

$$I_G(p) = 1 - pi^2 \quad (4)$$

Many authors in literature have sought ways to improve the classification accuracy of decision trees. Vijayshree etal. (2016) implemented a system that used support vector machine to improve the classification accuracy of decision trees. In their design, support vector machine was used to monitor user behavior while decision tree was used to determine whether user behavior was normal or fraudulent.

Sahin and Duman (2011) also applied support vector machines and decision tree in detecting credit card fraud. Their system used support vector machine to monitor each account, with identification based on suspicion score produced by the system and flags each transaction as normal or fraudulent transaction. Their system showed improved classification accuracy over normal decision tree.

Kaur & Gurm (2015) proposed the optimizing of Classification and Regression Trees with Genetic Algorithm to improve the classification accuracy of CART. Their work did not show details of the implementation.

Perner (2001) observed that classification accuracy of decision tree algorithm can be improved by using an appropriate feature pre-selection phase for the learning algorithm, thereby reducing the number of features used in building the classifier. They proved that classification accuracy can be improved if feature pre-selection is added in the learning algorithm.

2. METHODOLOGY OF THE PROPOSED MODEL

In our approach, synthetic data is generated from sample of real data from an anonymous bank in Nigeria. Using *RtoSynR* model, parameters which include cumulative distribution function of a variable distribution, mean of a distribution, variance, covariances and correlation are extracted from the real data using appropriate software and data analysis tools. The parameters are used to model the synthetic data. The generated synthetic data is joined with the sample of real dataset to form the Real-to-Synthetic Real (*RtoSynR*) dataset which is used to train a CART algorithm using information gain and gini impurity as splitting functions as well as random variable selection. The system fully followed the steps of *RtoSynR* modeling and implementation.

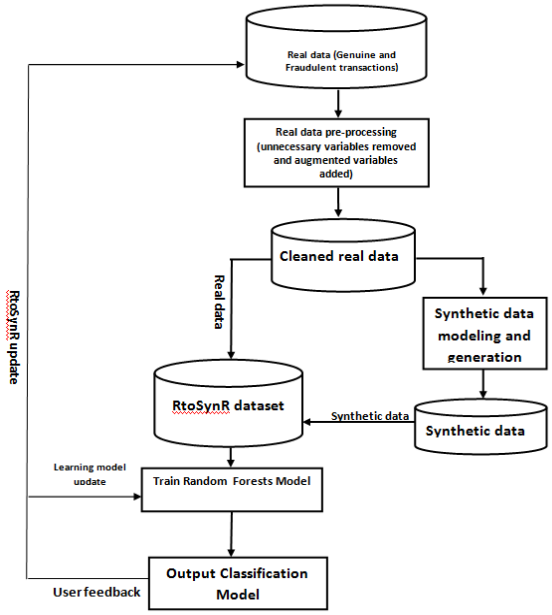
2.1. Architecture of the *RtoSynR* Model

Figure 1 shows the architecture of the RtoSynR model. As shown in the figure, the generated synthetic data is combined with the cleaned real data to form the RtoSynR dataset which is used to train Classification and Regression Tree models. Once deployed, the model now relies on customers' fraud feedback reports and fraud investigator analysis report to learn new fraud patterns and solve the problem of class imbalance and concept drift in the system. The feedback from the customers and the fraud analysis experts are fed into the real data database for simulation of more records of newly observed fraud pattern. It is also fed into the system training where it is combined with the synthetic data and used for the update of the learning algorithm.

2.2. General Algorithm of RtoSynR Model

Table 1 shows the algorithm of the RtoSynR model. The input to the algorithm is a sample real transaction data represented with $T_{Nr.}$ The algorithm is an open ended implementation that loops once it gets to the last step in the algorithm. The other parameters are explained in the next paragraph.

T_{Nr} = Real dataset, T_{Ns} = Synthetic dataset, T_N = Training set, T_{Ntrn} = Training dataset, T_{Ntst} = Testing dataset, $Trxn_{In}$ = Incoming transaction, $Trxn_{Ld}$ = New labeled transaction data (classified data), $T_{NrUpdate}$ = New update real data, $Trxn_{Ldf}$ = Class feedback from customers, $\bowtie$ = Join operator, *New*_T_{Ns} = New Synthetic dataset, New_Parameters = New parameters extracted from class feedback from customers.



F**igure1.** *The RtoSynR architecture*

Table 1: The RtoSynR algorithm

Step 1 {Generate training data}

Parameters = Call Extractor (T_{Nr})
[extract parameters from the input real data]

T_{Ns} = Call Simulator (Parameters)
[generate synthetic data with the parameters]

$T_N \leftarrow T_{Nr} \bowtie T_{Ns}$
[combine the real and synthetic data to form the training set T_N]

T_{Ntrn}, $T_{Ntst} \leftarrow T_N$
[Split the training set (T_N) into training dataset (T_{Ntrn}) and testing dataset (T_{Ntst})]

Step 2 {Train and test the model}

Classifier = Call Trainer (T_{Ntrn})
[produce a set of classifiers with the training set]

Testing Classifier ← Classifier (T_{Ntst})
[Test the generated Classifier with the testing set]

Step 3 {Classify incoming transactions}

$Trxn_{Ld}$ = Call Classifier ($Trxn_{In}$)
[classify incoming transactions into genuine or fraudulent class forming a new labeled data]

{System Update}

Step 4 {Real transaction data Update}

Update: $Trxn_{Ld\,f} \leftarrow$ Feedback ($Trxn_{Ld}$)
[receive class feedback from customers and fraud investigators]

$T_{NrUpdate} \leftarrow Trxn_{Ld} \bowtie Trxn_{Ldf}$
[form an update real labeled data by joining the newly classified data with the feedback from customers]

Update: $T_{Nr} \leftarrow T_{NrUpdate}$
[update the real training data with the update real data]

Step 5 JMP TO Step 7: If there is no change in transaction pattern ($Trxn_{Ldf}$ does not contain new patterns that need more samples); else Goto Step 6

Step 6 {Synthetic transaction data Update}

Update: New_Parameters = Call Extractor ($Trxn_{Ldf}$) [extract new parameters from the update real data]

*New*_T_{Ns} = Call Simulator (New_Parameters)
[generate new synthetic data]

Update: $T_{Ns} \leftarrow$ *New*_T_{Ns}

[update the synthetic data]

Step 7 {Training set Update}

Update: $T_N \leftarrow T_{Nr} \bowtie T_{Ns}$

[update the training set with the newly generated update real training data and synthetic real training data]

Update: T_{Ntrn}, $T_{Ntst} \leftarrow T_N$

[Split the training set (T_N) into training dataset (T_{Ntrn}) and testing dataset (T_{Ntst})]

Step 8 {Classifier Update}

Update: Classifier = Call Trainer (T_{Ntrn})
[update the classifier using the new update training set]

Update: Testing Classifier ← Classifier (T_{Ntst})

Step 9 JMP TO Step 3

3. DATASET DESCRIPTION

To implement our system, two debit card transaction datasets from an anonymous bank in Nigeria were used. The first was the genuine transaction dataset numbering 26,082 records. These were unreported transactions from 1000 customers (cardholders) that occurred in a 6 month period. The other was the 300 record fraudulent transactions dataset; which were reported fraudulent transactions from 300 customers in a 17 month period. The datasets had 10 features by default. 13 other features were derived from the table in order to augment the features as shown in table 2. Using the RtoSynR model, 6000 new records of fraudulent transaction were generated and combined with the real data to form 6300 records of the RtoSynR dataset.

Table2. *The dataset features*

Feature	Description	Data Type	Definition	Feature Type
1	Pos NO	Char	Point of Sale terminal number	Default
2	Acct NO	Char	Card/account number (anonymized value)	Default
3	Trans Date	Date/time	Transaction date and time (anonymized value)	Default
4	no_TRANS_MONTH	Integer	The number of transactions of that card in the last month	Default
5	Amount	Float	The amount of the transaction	Default
6	Merc Code	Integer	Merchant Category Code	Default
7	Card_Type	Integer	Card type (value missing)	Default
8	Exp_date	Date	Expiry date	Default
9	Merchant_Location	Integer	Zip code of the location of the merchant	Default
10	E-Commer	Integer	0 (No i.e. the transaction was not done on the internet via a payment gateway but rather with PoS) or 1 (Yes i.e. the transaction was done on the internet using a payment gateway)	Default
D1	Tot_Amt_Day	Float	The total amount of transaction for this card for the same day	Derived
D2	Num_Tran_Day	Integer	The number of transactions of this card for that same day	Derived
D3	Fail_Day	Integer	The failure number of transactions for this card in the same day	Derived
D4	Avg_Amt_Day	Float	The average transaction amount of this card for that same day	Derived
D5	Fail_7Day	Integer	The failure number of transactions of this card in the previous 7 days	Derived
D6	Avg_Amt_7Day	Float	The average transaction amount of this card in the previous 7 days	Derived
D7	Tot_Amt_30Day	Float	The total amount of transaction of this card in the previous 30 days	Derived
D8	Avg_Amt_30Day	Float	The average transaction amount of this card in the previous 30 days	Derived
D9	Num_Tran_Mer_Day	Integer	The number of transaction at the same merchant same day	Derived
D10	Num_Tran_Mer_30Day	Integer	The number of transactions at the same merchant in the previous 30 days	Derived
D11	Amt_Last_Trans	Float	Amount of the last transaction	Derived
D12	Dist_Dest	Integer	Distance between the transaction locations of the current and last transactions (Km)	Derived
D13	Time_Last_Trans	Float	Time elapsed since the last transaction (Minutes)	Derived

4. IMPLEMENTATION AND RESULTS

During the implementation of the system, the R console in the R language was used in the modeling and generating of the synthetic data. SPM 8.2 and WEKA were used in building the models and analyzing their performance. To test our system, the real dataset and the *RtoSynR* dataset were separately used to build CART models with test proportions of 0.1, 0.2, 0.3, 0.4, and 0.5 respectively. On SPM 8.2, models were built with information gain and gini impurity as splitting functions for the given test proportions. Similarly, on WEKA, models were built with REPTree which also used information gain as splitting function and Random Tree which randomly selects attributes for splitting. The obtained results were compared for the two datasets (*RtoSynR* dataset and real dataset). The results generated include the confusion matrix which shows the misclassification information of the models, and the testing metrics which include the Accuracy, Specificity, Recall, Precision and F1 Statistics. Also the error metrics were also computed for the datasets and test proportions. In all tests conducted for all test proportions, models built with *RtoSynR* datasets performed better than those built with real dataset, returning fewer misclassifications, improved accuracy metrics and lower error rates.

Table 3 shows the accuracy metrics of the datasets using information gain and gini impurity as splitting functions; while tables 4 and 5 show the confusion matrix for the datasets using gini impurity and information gain respectively. The total misclassifications were reduced from 7 misclassifications for real dataset to 3 misclassifications for *RtoSynR* dataset.

Similarly, tables 6 and 7 show the error metrics for the datasets using random tree and REPTree features in WEKA. Generally, as shown in the tables, *RtoSynR* dataset returned lower error values compared to real dataset. The result was particularly more significant in random tree than REPTree.

Table3. *Result of training with Real and RtoSynR datasets using CART algorithm, tested with varying test data proportions*

Test proportion	Test parameter	Split function			
		Gini		Information gain	
		Real dataset	*RtoSynR dataset*	*Real dataset*	*RtoSynR dataset*
0.1	Specificity	100.00%	100.00%	99.96%	100.00%
	Sensitivity/Recall	100.00%	100.00%	100.00%	100.00%
	Precision	100.00%	100.00%	96.55%	100.00%
	F1 Satistics	100.00%	100.00%	98.25%	100.00%
	Accuracy	100.00%	100.00%	99.96%	100.00%
0.2	Specificity	100.00%	100.00%	99.98%	100.00%
	Sensitivity/Recall	100.00%	100.00%	100.00%	100.00%
	Precision	100.00%	100.00%	98.36%	100.00%
	F1 Satistics	100.00%	100.00%	99.17%	100.00%
	Accuracy	100.00%	100.00%	99.98%	100.00%
0.3	Specificity	100.00%	100.00%	99.99%	100.00%
	Sensitivity/Recall	100.00%	100.00%	100.00%	100.00%
	Precision	100.00%	100.00%	98.98%	100.00%
	F1 Satistics	100.00%	100.00%	99.49%	100.00%
	Accuracy	100.00%	100.00%	99.99%	100.00%
0.4	Specificity	99.99%	100.00%	99.99%	99.99%
	Sensitivity/Recall	100.00%	100.00%	100.00%	100.00%
	Precision	99.24%	100.00%	99.24%	99.96%
	F1 Satistics	99.62%	100.00%	99.62%	99.98%
	Accuracy	99.99%	100.00%	99.99%	99.99%
0.5	Specificity	99.99%	99.99%	99.99%	99.99%
	Sensitivity/Recall	100.00%	100.00%	100.00%	99.93%
	Precision	99.34%	99.34%	99.34%	99.97%
	F1 Satistics	99.67%	99.67%	99.67%	99.95%
	Accuracy	99.99%	99.99%	99.99%	99.98%

Table3. *Confusion matrix of training with Real and RtoSynR datasets using CART algorithm and Gini impurity with varying test data proportions*

		Real dataset			*RtoSynR dataset*		
		Total Class Actual	*Predicted Classes*		Total Class Actual	*Predicted Classes*	
Test proportion	Parameter		Class 0	Class 1		Class 0	Class 1
0.1	Class 0	2,636	2,636	0	2,636	2,636	0
	Class 1	28	0	28	583	0	583
	Total:	2,664	2636	28	3219	2636	583
0.2	Class 0	5,257	5,257	0	5,257	5,257	0
	Class 1	60	0	60	1,248	0	1,248
	Total:	5,317	5257	60	6505	5257	1248
0.3	Class 0	7,758	7,758	0	7,758	7,758	0
	Class 1	97	0	97	1,863	0	1,863
	Total:	7,855	7758	97	9621	7758	1863
0.4	Class 0	10,422	10,421	1	10,422	10,422	0
	Class 1	130	0	130	2,478	0	2,478
	Total:	10,552	10421	131	12900	10422	2478
0.5	Class 0	13,013	13,012	1	13,013	13,013	0
	Class 1	151	0	151	3,064	1	3,063
	Total:	13,164	13012	152	16077	13014	3063

Table5. *Confusion matrix of training with Real and RtoSynR datasets using CART algorithm and Information gain with varying test data proportions*

		Real dataset			*RtoSynR dataset*		
		Total Class Actual	*Predicted Classes*		Total Class Actual	*Predicted Classes*	
Test proportion	Parameter		Class 0	Class 1		Class 0	Class 1
0.1	Class 0	2,636	2,635	1	2,636	2,636	0
	Class 1	28	0	28	583	0	583
	Total:	2,664	2635	29	3,219	2636	583
0.2	Class 0	5,257	5,256	1	5,257	5,257	0
	Class 1	60	0	60	1,248	0	1,248
	Total:	5,317	;25256	61	6,505	5257	1248
0.3	Class 0	7,758	7,757	1	7,758	7,758	0
	Class 1	97	0	97	1,863	0	1,863
	Total:	7,855	7757	98	9,621	7758	1863
0.4	Class 0	10,422	10,421	1	10,422	10,421	1
	Class 1	130	0	130	2,478	0	2,478
	Total:	10,552	10421	131	12,900	10421	2479
0.5	Class 0	13,013	13,012	1	13,013	13,014	1
	Class 1	151	0	151	3,064	0	3,062
	Total:	13,164	13012	152	16,077	13014	3063

Table6. *Error metrics of RtoSynR dataset compared with real dataset using random tree on WEKA*

Test proportion	Dataset	Number of instance	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
0.1	*RtosynR*	**3238**	**1**	**0.0001**	**0.0001**	**0.0319 %**	**0.0288 %**
	Real	2638	0.9836	0.0004	0.0195	1.6616 %	18.0669 %
0.2	*RtosynR*	**6476**	**0.998**	**0.0007**	**0.0249**	**0.2338 %**	**6.2777 %**
	Real	5276	0.9815	0.0004	0.0195	1.7621 %	19.5213 %
0.3	*RtosynR*	**9715**	**0.9974**	**0.0008**	**0.0287**	**0.2623 %**	**7.2289 %**
	Real	7915	0.9882	0.0003	0.0159	1.1454 %	15.512 %
0.4	*RtosynR*	**12953**	**0.9998**	**0.0002**	**0.0088**	**0.057 %**	**2.2014 %**
	Real	10553	0.9912	0.0002	0.0138	0.8487 %	13.2597 %
0.5	*RtosynR*	**16191**	**0.9965**	**0.0011**	**0.0333**	**0.3547 %**	**8.4016 %**
	Real	13191	0.9774	0.0005	0.023	2.3602 %	21.5146 %

Table7. *Error metrics of RtoSynR dataset compared with real dataset using REPtree on WEKA*

Test proportion	Dataset	Number of instance	Correlation coefficient	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
0.1	*RtosynR*	**3238**	**0.9981**	**0.0008**	**0.0249**	**0.2479 %**	**6.1886 %**
	Real	2638	1	0.0001	0.0001	0.553 %	0.118 %
0.2	*RtosynR*	**6476**	**0.998**	**0.0008**	**0.0248**	**0.2586 %**	**6.2772 %**
	Real	5276	1	0.0001	0.0001	0.6609 %	0.1436 %
0.3	*RtosynR*	**9715**	**0.999**	**0.0005**	**0.0176**	**0.1544 %**	**4.4264 %**
	Real	7915	1	0.0002	0.0005	0.9608 %	0.4962 %
0.4	*RtosynR*	**12953**	**0.9995**	**0.0004**	**0.0124**	**0.1302 %**	**3.1134 %**
	Real	10553	0.9912	0.0005	0.0138	2.1878 %	13.2516 %
0.5	*RtosynR*	**16191**	**0.9986**	**0.0007**	**0.0208**	**0.236 %**	**5.2383 %**
	Real	13191	0.9934	0.0002	0.0123	0.6743 %	11.5 %

5. CONCLUSION

The use of *RtoSynR* dataset improved the classification accuracy of CART models, reducing the classification error and the total number of misclassified instances to 3 from 7. Generally, in all tests carried out with various test proportions and splitting functions for Classification and Regression Trees, *RtoSynR* performed better than real dataset in debit card fraud detection.

REFERENCES

[1] Alpaydin, E. (2010). *Introduction to Machine Learning.* Massachusetts Institute of Technology, United States.

[2] Andrea, D. P., Reid, A. J., Olivier, C. S., Nitesh, V. C., and Gianluca, B. (2014). Using HDDT to Avoid Instances Propagation in Unbalanced and Evolving Data Streams in Neural Networks. *Paper Presented at the IEEE International Joint Conference on Neural Networks (IJCNN) 2014.* Retrieved July 7th, 2017, from http://ieeexplore.ieee.org/document/6889638/

[3] Barry, D. V., and N. Padraic (2013). *Decision Trees for Analytics Using SAS Enterprise Miner.* SAS Institute.

[4] Bhumika, G. (2017). Analysis of Various Decision Tree Algorithms for Classification in Data Mining. *International Journal of Computer Applications, 163*(8).

[5] Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science, 17*(3), 235-255.

[6] Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone (1984). *Classification and Regression Trees. Monterey.* California: Wadsworth and Brooks/Cole Advanced Books and Software.

[7] Fleishman, A. I. (1978). A Method For Simulating Non-normal Distributions. *Psychometrika*, 521-532. doi:10.1007/BF02293811

[8] Haibo, H., and Edwardo, A. G. (2009). Learning from Imbalanced Data. *Knowledge and Data Engineering, 21*(9), 1263–1284.

[9] Jatinder, K., and S. G. Jasmeet (2015). Optimizing the Accuracy of CART algorithm by Using Genetic Algorithm. *International Journal of Computer Science Trends and Technology (IJCST), 3*(4).

[10] Jon, T., and Sriganesh, M. (2008). Real-Time Credit Card Fraud Detection Using Computational Intelligence. *Expert Systems with Applications , 35*(4), 1721–1732.

[11] Kaur, J., & Gurm, J. S. (2015). Optimizing the Accuracy of CART Algorithm by Using Genetic Algorithm. *International Journal of Computer Science Trends and Technology, 3*(4).

[12] Lepoivre, M. R., Avanzini, C. O., Bignon, G., Legendre, L., & Piwele, A. K. (2016). Credit Card Fraud Detection with Unsupervised Algorithm. *Journal of Advances in Information Technology, 7*(1).

[13] Lopez-Rojas, E. A., and Axelsson, S. (2012). Money Laundering Detection Using Synthetic Data. *Paper presented in the 27th Workshop of Swedish Artificial Intelligence Society (SAIS) (2012)*, 33-40. Retrieved December 6th, 2016, from https://pdfs.semanticscholar.org/ 39f5/c1f3bfed9b85b7a126e6e4da35 87d3f161 60.pdf

[14] Masa, P., and Kocza. (2006). Finding Optimal Decision Trees. In M. Kłopotek, S. Wierzchoń, and K. Trojanowski (Ed.), *Proceedings of the International Intelligent Information Processing and Web Mining.35*, 173-181. Ustron, Poland: Springer. doi:https://doi.org/10.1007/3-540-33521-8_17

[15] Meshram, P. L., & Bhanarkar, P. (2012). Credit and ATM Card Fraud Detection Using Genetic Approach. *International Journal of Engineering Research & Technology, 1*(10).

[16] Neha, P., Roy, W., and Kalyan, V. (2016). The Synthetic Data Vault. *proceedings of th2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA).* doi:10.1109/DSAA.2016.49

[17] Perner, P. (2001). Improving the Accuracy of Decision Tree Induction by Feature Pre-Selection. *Applied Artificial Intelligence, 15*(8), 747-760.

[18] Quinlan, J. R. (1996). Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research, 4*, 77-90.

[19] Sahin, Y., and Duman, E. (2011). Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. *Proceedings of the International MultiConference of Engineers and Computer Scientist 2011*. Honkong. Retrieved February 10th, 2018, from www.iaeng.org/publication/IMECS2011/IMECS2011_ pp442-447.pdf

[20] Sonepat, S. H., and Sonepat, R. H. (2011). Analysis on Credit Card Fraud Detection Methods. *International Journal of Computer Trends and Technology (IJCTT), 8*(1).

[21] Tsymbal, A. (2004). *The problem of concept drift: definitions and related work.* Trinity College, Computer Science. Dublin: Citeseer. Retrieved January 3rd, 2018, from http://citeseerx.ist.psu.edu/viewdoc/ download?doi=10.1.1.58.9085&rep=rep1&type=pdf

[22] Vijayshree, B. N., Poonam, S. K., Dipali, V., Kunal, W., and Bhagyashree, P. D. (2016). Fraudulent Detection in Credit Card System Using SVM and Decision Tree. *International Journal of Scientific Development and Research (IJSDR), 1*(5).

[23] Webb, G. I., Lee, L. K., Goethais, B., & Petitjean, F. (2018). Analyzing Concept Drift and Shift from Sample Data. *Data Mining and Knowledge Discovery, 32*(5). Zhang, X., Yanwei, F., Andi, Z., Leonid, S., and Gady, A. (2015). *Learning Classifiers from Synthetic Data Using a Multichannel Autoencoder.* Retrieved December 10th, 2016, from https://pdfs.semanticscholar.org/2071/7f1cb12ab208458c0f2505b2 37d8f061f97a.pdf

Citation: *Nwogu E. R, et.al., (2019). "*On Reducing Misclassifications and Error on CART Models using Rtosynr Dataset for Improved Debit Card Fraud Detection*". International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 6(4), pp.7-15. http://dx.doi.org/10.20431/2349-4859.0604002*