DETECTION OF FRAUDULENT FINANCIAL STATEMENTS USING THE HYBRID DATA MINING APPROACH

A PROJECT REPORT

Submitted by

ARAVINDS B R
14ITR007

KISHORE N
14ITR041

MANISHA B
14ITR046

in partial fulfilment of the requirements

for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMMUNICATION AND COMPUTER SCIENCES

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

APRIL 2018

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638060

APRIL 2018

BONAFIDE CERTIFICATE

This is to certify that the Project Report entitled **DETECTION OF FRAUDULENT FINANCIAL STATEMENTS USING THE HYBRID DATA MINING APPROACH** is bonafide record of project work done by **ARAVINDS B R** (14ITR007), **KISHORE N** (14ITR041) and **MANISHA B** (14ITR046) in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology of Anna University, Chennai during the year 2017 - 2018.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva voce examination held on ___________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF INFORMATION TECHNOLOGY

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638060

APRIL 2018

DECLARATION

We affirm that the Project Report entitled **DETECTION OF FRAUDULENT FINANCIAL STATEMENTS USING THE HYBRID DATA MINING APPROACH** being submitted in partial fulfillment of the requirements for the award of Bachelor of Technology is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

ARAVINDS B R
14ITR007

Date:

KISHORE N
14ITR041

MANISHA B
14ITR046

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: Name and Signature of the Supervisor with seal

ABSTRACT

Financial statements are basic documents represent financial status of a company. Fraudulent financial statements are intentional and illegal acts that result in misleading financial statements or misleading financial disclosure. It is important to detect the fraudulent financial statement prior to its occurrence and prevent company's financial loss. The purpose of the project is to construct fraudulent financial statement detection model which identifies whether the financial statement of an organization is either fraud or not fraud. The percentage of data considered as training data, test data and validation data in Random Forest,Artificial Neural Network(ANN), Gradient Boosting Machine(GBM) and Generalized Linear Model(GLM) will differ. The detection of fraudulent financial statement of an organization will reduce the loss incurred due to the fraud made with the help of financial statement.

The credit card dataset is given as the input for all the implemented methods. Initially, variable selection process takes place where along with 28 variables amount and class are also used. Random forest or random decision forest method is used which is an ensemble learning method for classification, regression and other tasks. In the next stage, in order to construct fraudulent financial statement detection models the above implemented Artificial Neural Network, Gradient Boosting Machine,Generalized Linear Model are used for the selection of more optimized result within these models.

.

ACKNOWLEDGEMENT

First and foremost, we acknowledge the abundant grace and presence of Almighty throughout different phases of the project and its successful completion.

We wish to express our extreme gratefulness to our beloved Correspondent **Thiru.A.VENKATACHALAM B.Sc.,** and all the trust members of Kongu Vellalar Institute of Technology Trust for providing all the necessary facilities to complete the project successfully.

We express our deep sense of gratitude to our beloved Principal **Prof.S.KUPPUSWAMI B.E., M.Sc(Engg)., Dr.Ing(France).,**for providing us an opportunity to complete the project.

We express our gratitude and heartfelt sincere thanks to **Dr.S.VARADHAGANAPATHY M.S., M.E., Ph.D.,** Head, Department of Information Technology, for his valuable suggestions.

We are thankful to our project coordinators **Dr.C.NALINI M.E., Ph.D.,** and **Mrs.S.ANITHA M.E.,** for their valuable guidance and support to complete our project successfully.

We are highly indebted to **Dr.P.SURESH, M.E., Ph.D.,** Department of Information Technology for his valuable supervision and advice for the fruitful completion of the project.

We are very much thankful to our Department Faculty members for their valuable support and suggestions.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE No.
	ABSTRACT	**iv**
	LIST OF FIGURES	**viii**
	LIST OF ABBREVIATIONS	**ix**
1	**INTRODUCTION**	**1**
	1.1 DATA MINING	1
	1.1.1 The Scope of Data mining	1
	1.1.2 Data mining process	2
	1.1.3 Classification	2
	1.1.4 Prediction	2
	1.2 OBJECTIVE	3
2	**SYSTEM ANALYSIS**	**4**
	2.1 LITERATURE REVIEW	4
	2.2 SUMMARY	5
3	**PROBLEM DEFINITION**	**6**
	3.1 PROBLEM STATEMENT	6
	3.2 PROPOSED METHOD	7
4	**SYSTEM REQUITEMENTS**	**8**
	4.1 SOFTWARE REQUIREMENTS	8
	4.2 HARDWARE REQUIREMENTS	8

	4.3 PACKAGE DESCRIPTION	8
5	**SYSTEM DESIGN**	**9**
	5.1 OVERVIEW	9
	5.2 MODULE DESCRIPTION	10
	5.2.1 Random Forest	10
	5.2.2 Gradient Boosting Method	10
	5.2.3 Generalized Linear Model	11
	5.2.4 Artificial Neural Network	11
6	**SYSTEM IMPLEMENTATION**	**13**
	6.1 DATASET	13
	6.2 RESULT ANALYSIS	14
7	**CONCLUSION AND FUTURE ENHANCEMENT**	**17**
	7.1 CONCLUSION	17
	7.2 FUTURE ENHANCEMENT	17
	APPENDICES	**18**
	APPENDIX 1 - CODING	18
	APPENDIX 2 – SNAPSHOTS	34
	REFERENCES	**36**

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	Research design and procedure	7
5.1	Flow Diagram	9
5.2	Processing in an Artificial Neuron	10
6.1	Credit Card Dataset	14
6.2	Training Data Result	15
6.3	Test Data Result	16
A.2.1	Variable Selection	34
A.2.2	Accuracy of ANN	34
A.2.3	GBM Result	35

LIST OF ABBREVIATIONS

CHAID	-	Chi-square Automatic Interaction Detector
RF	-	Random Forest
BBN	-	Bayesian Belief Network
SVM	-	Support Vector Machine
GBM	-	Gradient Boosting Machine
GLM	-	Generalized Linear Model
ANN	-	Artificial Neural Network
CART	-	Classification And Regression Trees

CHAPTER 1

INTRODUCTION

1.1 DATA MINING

Data mining is a process used by companies to turn raw data into useful information. By using software to look for patterns in large batches of data, businesses can learn more about their customers and develop more effective marketing strategies as well as increase sales and decrease costs. Data mining depends on effective data collection and warehousing.

1.1.1 The Scope of Data mining

Data mining derives its name from the similarities between searching for valuable business information in a large database for example, finding linked products in gigabytes of store scanner data and mining a mountain for a vein of valuable ore. Both processes require either shifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

- **Automated prediction of trends and behaviours :** Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data quickly.
- **Automated discovery of previously unknown patterns :** Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together.

1.1.2 Data mining process

The data mining process breaks down into some steps. First, organizations collect data and load it into their data warehouses. Next, they store and manage the data. Business analysts, management teams and information technology professionals access the data and determine how they want to organize it. Then, application software sorts the data based on the user's results, and finally, the end user presents the data in the form of graph

1.1.3 Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks. A classification task begins with a data set in which the class assignments are known. For example, a classification model that predicts credit risk could be developed based on observed data for many loan applicants over a period of time. In addition to the historical credit rating, the data might track employment history, home ownership or rental, years of residence, number and type of investments, and so on. Credit rating would be the target, the other attributes would be the predictors, and the data for each customer would constitute a case. Classifications are discrete and do not imply order. Continuous, floating-point values would indicate a numerical, rather than a categorical, target. A predictive model with a numerical target uses a regression algorithm, not a classification algorithm.

1.1.4 Prediction

Predictive analytics encompasses a variety of statistical techniques from predictive modeling, machine learning, and data mining that analyze current and historical facts to make predictions about future or otherwise unknown events. In business, predictive models exploit patterns found in historical and transactional data to identify risks and opportunities. Models capture relationships among many factors to allow assessment of risk or potential associated with a particular set of conditions, guiding decision making for candidate transactions. The defining functional effect of these technical approaches is that predictive analytics provides a predictive score (probability) for each individual (customer, employee, healthcare patient, vehicle, component, machine, or other organizational unit) in order to determine, inform, or influence organizational processes that pertain across large numbers of individuals, such as in marketing, credit risk assessment, fraud detection, manufacturing, healthcare, and government operations including law enforcement. The Predictor resource mines data collected in the Data Warehouse. The Data Warehouse database can store all site data, including user profile data, click-history data, and transaction data. Although the Data Warehouse database is quite large, the capabilities of the Predictor resource can efficiently mine the data.

1.2 OBJECTIVE

The objective of this project is to detect the fraudulent financial statements of an organization using an effective and accurate fraudulent financial statement detection model. Financial statements are company's basic documents that reflect its financial status. The financial statement is the main basis for decision making on the part of a vast number of investors, creditors and other persons in need of accounting information, as well as a concrete expression of business performance, financial status and the social responsibility of listed companies. The process strives to bring out the methodology and implementation of these classification techniques and stress upon the results and conclusion induced on the basis of accuracy and time complexity.

CHAPTER 2

SYSTEM ANALYSIS

2.1 LITERATURE REVIEW

In research studies Lenard and Alam, (2009) proposed that the analysis of ratios is chosen as one of the methods to determine fraud. Financial difficulties may be motivation for managers to engage in fraudulent activities. According to Ravisankar et al. (2011), the higher levels of debt may increase the probability of the fraudulent financial statements too.

Wei Zhou (2011) proposed the Detecting evolutionary financial statement fraud in which data mining is used widely in financial statements especially when new schemes of financial statement fraud adapt to the detection techniques. Therefore they concluded by specifying the future research is needed to design the active discovery module that is both effective and efficient. Furthermore, although they have suggested the response surface method to extract the domain knowledge and to adaptively learn the changes from fraudsters, there may exist alternatives that have equal or better performance than the RSM.

Wang et al., (2012) proposed the hybrid system consists of machine learning and rule-based classifiers. For the machine learning classifier, we investigate a variety of lexical, syntactic and knowledge-based features, and show how much these features contribute to the performance of the classifier through experiments. Na et al. (2012) proposed a rule-based system for polarity classification. Clause-level sentiment classification algorithm is developed and applied to fraudulent reviews on a discussion forum.

2.2 SUMMARY

The literature survey helps to understand more about the trending data mining model for fraudulent financial statement detection rather than the conventional models such as CHAID and CART. The literary references indicate that the usage of the financial ratios for determining the fraudulent statements of financial reports is a convenient and straightforward means. However, the problem of interpreting the result interpretation arises, i.e. what value of financial ratio indicates that the reports are fraudulent.

CHAPTER 3

PROBLEM DEFINITION

3.1 PROBLEM STATEMENT

The regression analysis method was initially used in organizations to detect the fraudulent financial statements. But the regression analysis method causes a number of disadvantages in terms of judgment and its error rate is relatively high. Regression analysis is a form of predictive modeling technique which investigates the relationship between a dependent and independent variable. The judgment accuracy rates of using data mining techniques to detect fraudulent financial statements of any organization vary, and the construction of the model is neither complete nor perfect. There are four elements of fraud that are found in the case of financial statements. They are,

1) serious erroneous expressions of the nature of transactions
2) knowingly violating rules
3) the victim accepting a misstatement as fact
4) damage due to financial losses caused by the above three situations.

Figure 3.1 shows the working of the model where the original data of the year before the year of fraud is given as the input and then the variables are classified as financial and non financial variables and then a two-stage statistical treatment is followed. Data mining techniques are then used to create the detection model for comparison of model accuracy. Studies applying DT techniques to detect fraudulent financial statements. The judgment accuracy rates of using data mining techniques to detect fraudulent financial statements vary and the construction of the model is neither complete nor perfect. As stated above, most studies only use 1–2 data mining techniques, without offering model comparison and most use one-stage statistical treatment to establish the detection model.

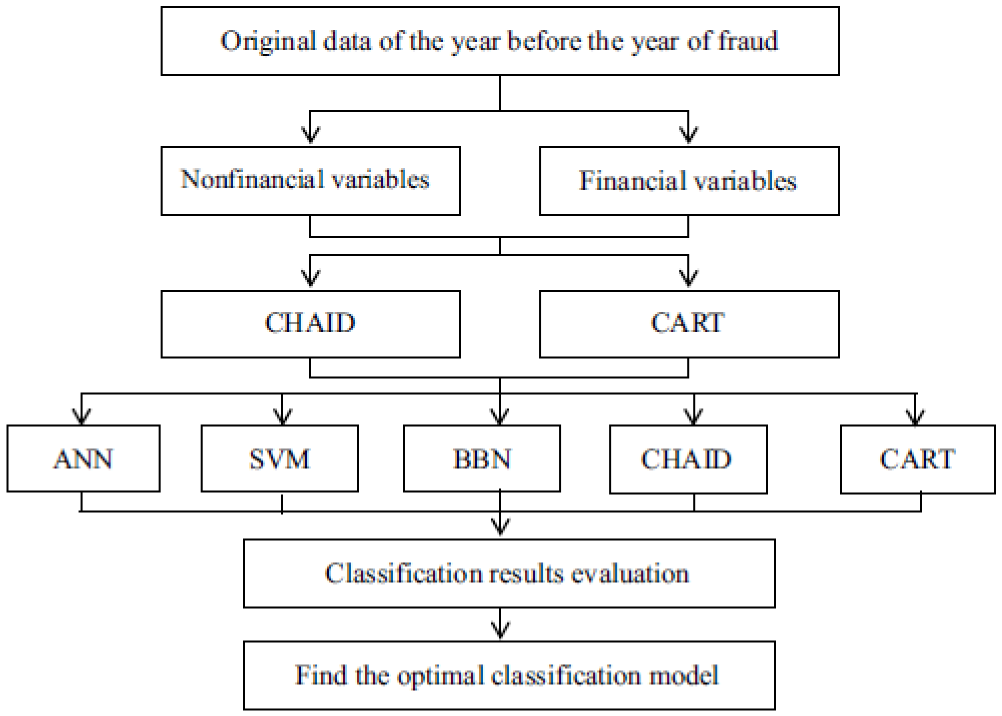

Original data of the year before the year of fraud
Nonfinancial variables
Financial variables
CHAID
CART
ANN
SVM
BBN
CHAID
CART
Classification results evaluation
Find the optimal classification model

Figure 3.1 Research design and procedure

3.2 PROPOSED METHOD

The proposed method to detect the fraudulent financial statements using the credit card dataset with various models such as Random Forest, Artificial Neural Network(ANN), Gradient Boosting Machine(GBM), Generalized Linear method(GLM). The purpose of the project is to construct a valid and rigorous fraudulent financial statement detection model. Analysis of financial details is one of those simple methods to identify frauds. The aim of the research is to distinguish financial details, the values of which could indicate the fraud in financial statements. Moreover, the logistic regression model of fraud detection in financial statements has been developed. In the first stage, for the variable selection process appropriate deviations are used with certain constrains. Random forest or random decision forest method is used which is an ensemble learning method for classification, regression and other tasks. In the second stage, in order to construct fraudulent financial statement detection models ANN, GBM, GLM are used for the selection of more optimized result within these models. Data mining tools allow enterprises to predict future trends.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

Operating system	:	Windows 8.1
Language	:	Python 3.6.4

4.2 HARDWARE REQUIREMENTS

Processor	:	Dual Core
Storage	:	100GB
RAM	:	4GB

4.3 PACKAGE DESCRIPTION

pandas	:	Provides data structures to work with structured, time series data.
seaborn	:	Library for making statistical graphics
H2O	:	Used for data modeling and general computing
matplotlib	:	Package publishes quality 2D graphics
numpy	:	General-purpose array-processing package

CHAPTER 5

SYSTEM DESIGN

5.1 OVERVIEW

In the existing system CHAID and CART were used. Since they are conventional algorithm the recent methods are used. They are Random Forest(RF), Artificial Neural Network(ANN), Gradient Boosting Machine(GBM) and Generalized Linear Model(GLM). The percentage of data to be used as training data, validation data and test data will differ for each and every method.

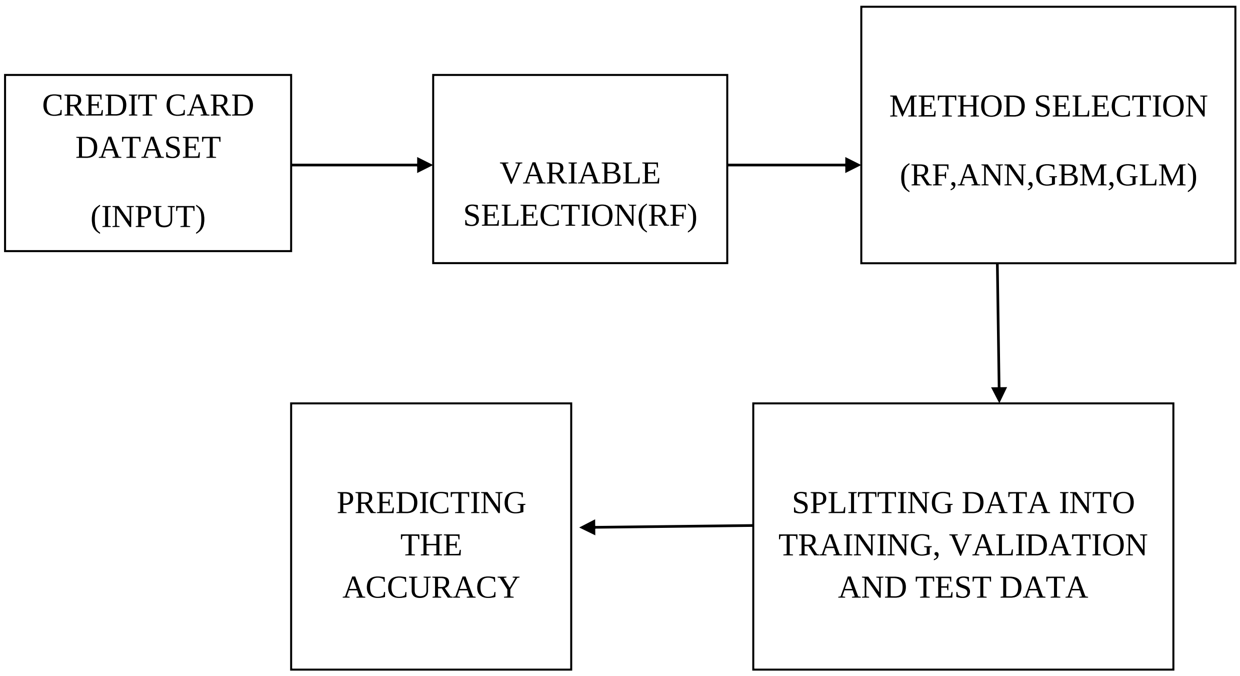


Figure 5.1 Flow Diagram

5.2 MODULE DESCRIPTION

5.2.1 Random Forest

Random forests is an ensemble learning method for classification and regression, that operate by constructing a multitude of decision tree at training time and outputting the class that is the mode of the class or mean prediction of the individual trees .Random forest uses full decision trees. It has low bias and high variance where variance can be defined as the error from the small fluctuations in the training data set and bias can be defined as the error from the spontaneous assumptions. The prediction describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and bagging.

Important points are,

i. Using out-of-bag error as an estimate of the generalization error.

ii. Measuring variable importance through permutation.

The algorithm working is explained as follows,

1. The input for the Random Forest Algorithm is the credit card dataset which consists of 30 variables along with the class and amount variable.
2. The data will be split into training and test data. Then, the Random Forest model is constructed and then the testing of data follows.
3. The confusion matrix is built from which the accuracy can be calculated.

5.2.2 Gradient Boosting Machine

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The trees will be built with the help of previously built trees.

The algorithm working is explained as follows,

1. The credit card dataset is given as the input.
2. The data will be split into three parts such as train, test and valid data.

3. In training data, the model is built. In test data, the output will be predicted with the help of training data. The validation data will be used to validate whether the given output is valid or not.

5.2.3 Generalized Linear Model

In statistics, the generalized linear model (GLM) is a flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution.

The algorithm working is explained as follows,

1. The input for the Generalized Linear Model is the credit card dataset which consists of 30 variables which excludes the class and amount variable.
2. The data will be split into three parts such as train, test and valid data.
3. The default number of trees is set to 50 initially and the learning rate is increased from the default value.
4. The default number of trees is further decreased and the learning rate is increased from the previous value.
5. In training data, the model is built. In test data, the output will be predicted with the help of training data. The validation data will be used to validate whether the given output is valid or not.

5.2.5 Artificial Neural Network

Neural computing refers to a pattern recognition methodology for machine learning. The resulting model from neural computing is often called an Artificial Neural Network (ANN) or a neural network. Figure 5.1 represents that the ANN is composed of multiple nodes such as $X_1,X_2 \ldots X_i$, The result of these operations is passed to other neurons.

The output at each node is called its activation or node value. Each link is associated with weight $W_{1j},W_{2j} \ldots W_{ij}$. ANNs are capable of learning, which takes place by altering weight values. Neural networks have been used in many business applications for pattern recognition, forecasting, prediction, and classification. Neural network computing is a key component of any data mining tool kit.

The algorithm working is explained as follows,

1. The credit card dataset is given as the input for Artificial Neural Network
2. The data will be split into three parts such as train, test and valid data.
3. Tuning is performed to reduce the error rate.
4. In training data, the model is built. In test data, the output will be predicted with the help of training data. The validation data will be used to validate whether the given output is valid or not.
5. The number of hidden layers are increased to predict the improved accuracy.

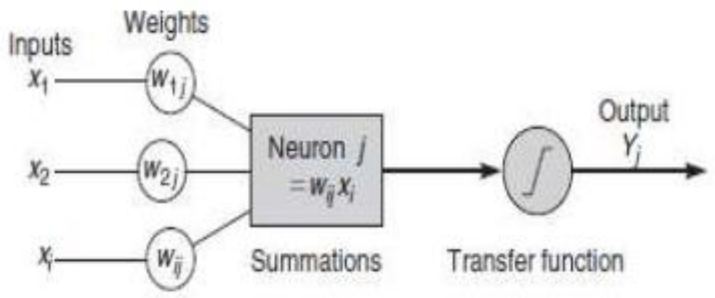


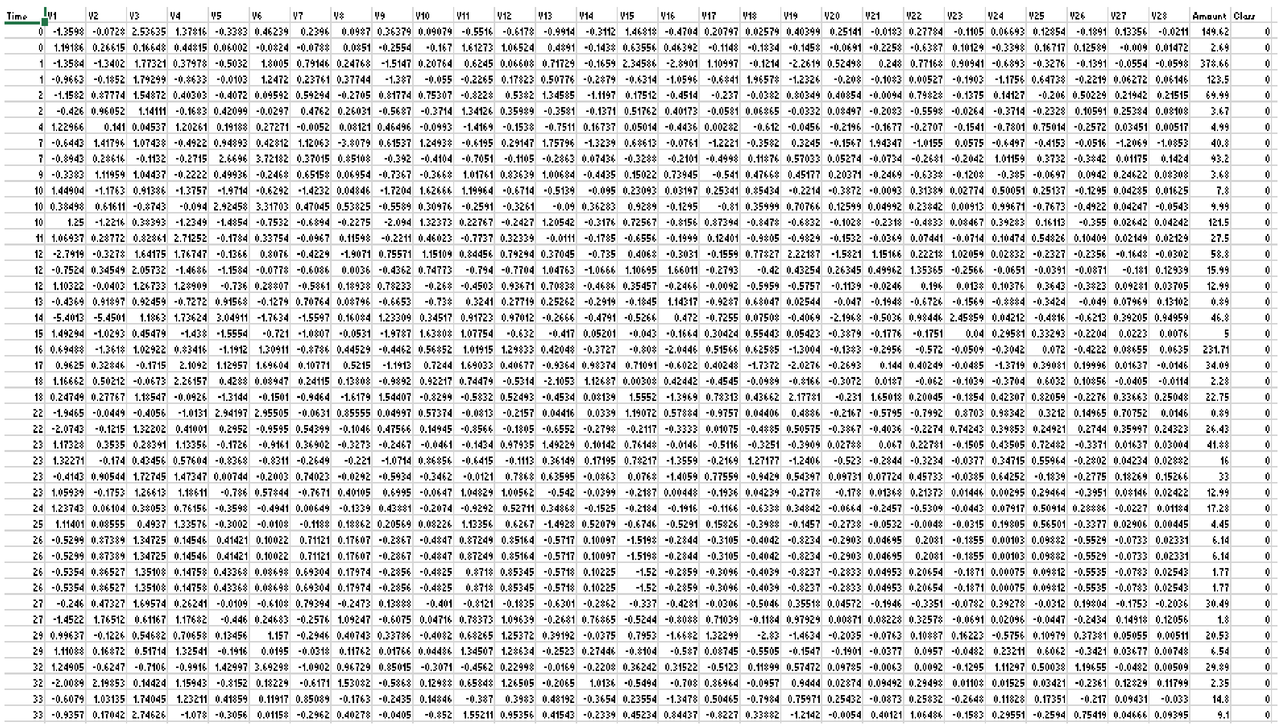
Figure 5.2 Processing in an Artificial Neuron

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 DATASET

The credit card details dataset is taken from the web resource www.kaggle.com. Figure 6.1 shows the sample details format from the www.kaggle.com. The credit card dataset consists of 30 variables in total. Since the credit card dataset deals with confidential information, the names of the attributes used in the credit card dataset is not revealed for the security purposes.



Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	-1.3598	-0.0728	2.53635	1.37816	-0.3383	0.46239	0.2396	0.0987	0.36379	0.09079	-0.5516	-0.6178	-0.9914	-0.3112	1.46818	-0.4704	0.20797	0.02579	0.40399	0.25141	-0.0183	0.27784	-0.1105	0.06693	0.12854	-0.1891	0.13356	-0.0211	149.62	0
0	1.19186	0.26615	0.16648	0.44815	0.06002	-0.0824	-0.0788	0.0851	-0.2554	-0.167	1.61273	1.06524	0.4891	-0.1438	0.63556	0.46392	-0.1148	-0.1834	-0.1458	-0.0691	-0.2258	-0.6387	0.10129	-0.3398	0.16717	0.12589	-0.009	0.01472	2.69	0
1	-1.3584	-1.3402	1.77321	0.37978	-0.5032	1.8005	0.79146	0.24768	-1.5147	0.20764	0.6245	0.06608	0.71729	-0.1659	2.34586	-2.8901	1.10997	-0.1214	-2.2619	0.52498	0.248	0.77168	0.90941	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
1	-0.9663	-0.1852	1.79299	-0.8633	-0.0103	1.2472	0.23761	0.37744	-1.387	-0.055	-0.2265	0.17823	0.50776	-0.2879	-0.6314	-1.0596	-0.6841	1.96578	-1.2326	-0.208	-0.1083	0.00527	-0.1903	-1.1756	0.64738	-0.2219	0.06272	0.06146	123.5	0
2	-1.1582	0.87774	1.54872	0.40303	-0.4072	0.09592	0.59294	-0.2705	0.81774	0.75307	-0.8228	0.5382	1.34585	-1.1197	0.17512	-0.4514	-0.237	-0.0382	0.80349	0.40854	-0.0094	0.79828	-0.1375	0.14127	-0.206	0.50229	0.21942	0.21515	69.99	0
2	-0.426	0.96052	1.14111	-0.1683	0.42099	-0.0297	0.4762	0.26031	-0.5687	-0.3714	1.34126	0.35989	-0.3581	-0.1371	0.51762	0.40173	-0.0581	0.06865	-0.0332	0.08497	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.10591	0.25384	0.08108	3.67	0
4	1.22966	0.141	0.04537	1.20261	0.19188	0.27271	-0.0052	0.08121	0.46496	-0.0993	-1.4169	-0.1538	-0.7511	0.16737	0.05014	-0.4436	0.00282	-0.612	-0.0456	-0.2196	-0.1677	-0.2707	-0.1541	-0.7801	0.75014	-0.2572	0.03451	0.00517	4.99	0
7	-0.6443	1.41796	1.07438	-0.4922	0.94893	0.42812	1.12063	-3.8079	0.61537	1.24938	-0.6195	0.29147	1.75796	-1.3239	0.68613	-0.0761	-1.2221	-0.3582	0.3245	-0.1567	1.94347	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
7	-0.8943	0.28616	-0.1132	-0.2715	2.6696	3.72182	0.37015	0.85108	-0.392	-0.4104	-0.7051	-0.1105	-0.2863	0.07436	-0.3288	-0.2101	-0.4998	0.11876	0.57033	0.05274	-0.0734	-0.2681	-0.2042	1.01159	0.3732	-0.3842	0.01175	0.1424	93.2	0
9	-0.3383	1.11959	1.04437	-0.2222	0.49936	-0.2468	0.65158	0.06954	-0.7367	-0.3668	1.01761	0.83639	1.00684	-0.4435	0.15022	0.73945	-0.541	0.47668	0.45177	0.20371	-0.2469	-0.6338	-0.1208	-0.385	-0.0697	0.0942	0.24622	0.08308	3.68	0
10	1.44904	-1.1763	0.91386	-1.3757	-1.9714	-0.6292	-1.4232	0.04846	-1.7204	1.62666	1.19964	-0.6714	-0.5139	-0.095	0.23093	0.03197	0.25341	0.85434	-0.2214	-0.3872	-0.0093	0.31389	0.02774	0.50051	0.25137	-0.1295	0.04285	0.01625	7.8	0
10	0.38498	0.61611	-0.8743	-0.094	2.92458	3.31703	0.47045	0.53825	-0.5589	0.30976	-0.2591	-0.3261	-0.09	0.36283	0.9289	-0.1295	-0.81	0.35999	0.70766	0.12599	0.04992	0.23842	0.00913	0.99671	-0.7673	-0.4922	0.04247	-0.0543	9.99	0
10	1.25	-1.2216	0.38393	-1.2349	-1.4854	-0.7532	-0.6894	-0.2275	-2.094	1.32373	0.22767	-0.2427	1.20542	-0.3176	0.72567	-0.8156	0.87394	-0.8478	-0.6832	-0.1028	-0.2318	-0.4833	0.08467	0.39283	0.16113	-0.355	0.02642	0.04242	121.5	0
11	1.06937	0.28772	0.82861	2.71252	-0.1784	0.33754	-0.0967	0.11598	-0.2211	0.46023	-0.7737	0.32339	-0.0111	-0.1785	-0.6556	-0.1999	0.12401	-0.9805	-0.9829	-0.1532	-0.0369	0.07441	-0.0714	0.10474	0.54826	0.10409	0.02149	0.02129	27.5	0
12	-2.7919	-0.3278	1.64175	1.76747	-0.1366	0.8076	-0.4229	-1.9071	0.75571	1.15109	0.84456	0.79294	0.37045	-0.735	0.4068	-0.3031	-0.1559	0.77827	2.22187	-1.5821	1.15166	0.22218	1.02059	0.02832	-0.2327	-0.2356	-0.1648	-0.0302	58.8	0
12	-0.7524	0.34549	2.05732	-1.4686	-1.1584	-0.0778	-0.6086	0.0036	-0.4362	0.74773	-0.794	-0.7704	1.04763	-1.0666	1.10695	1.66011	-0.2793	-0.42	0.43254	0.26345	0.49962	1.35365	-0.2566	-0.0651	-0.0391	-0.0871	-0.181	0.12939	15.99	0
12	1.10322	-0.0403	1.26733	1.28909	-0.736	0.28807	-0.5861	0.18938	0.78233	-0.268	-0.4503	0.93671	0.70838	-0.4686	0.35457	-0.2466	-0.0092	-0.5959	-0.5757	-0.1139	-0.0246	0.196	0.0138	0.10376	0.3643	-0.3823	0.09281	0.03705	12.99	0
13	-0.4369	0.91897	0.92459	-0.7272	0.91568	-0.1274	0.70764	0.08796	-0.6653	-0.738	0.3241	0.27719	0.25262	-0.2919	-0.1845	1.14317	-0.9287	0.68047	0.02544	-0.047	-0.1948	-0.6726	-0.1569	-0.8884	-0.3424	-0.049	0.07969	0.13102	0.89	0
14	-5.4013	-5.4501	1.1863	1.73624	3.04911	-1.7634	-1.5597	0.16084	1.23309	0.34517	0.91723	0.97012	-0.2666	-0.4791	-0.5266	0.472	-0.7255	0.07508	-0.4069	-2.1968	-0.5036	0.98446	2.45859	0.04212	-0.4816	-0.6213	0.39205	0.94959	46.8	0
15	1.49294	-1.0293	0.45479	-1.438	-1.5554	-0.721	-1.0807	-0.0531	-1.9787	1.63808	1.07754	-0.632	-0.417	0.05201	-0.043	-0.1664	0.30424	0.55443	0.05423	-0.3879	-0.1776	-0.1751	0.04	0.29581	0.33293	-0.2204	0.0223	0.0076	5	0
16	0.69488	-1.3618	1.02922	0.83416	-1.1912	1.30911	-0.8786	0.44529	-0.4462	0.56852	1.01915	1.29833	0.42048	-0.3727	-0.808	-2.0446	0.51566	0.62585	-1.3004	-0.1383	-0.2956	-0.572	-0.0509	-0.3042	0.072	-0.4222	0.08655	0.0635	231.71	0
17	0.9625	0.32846	-0.1715	2.1092	1.12957	1.69604	0.10771	0.5215	-1.1913	0.7244	1.69033	0.40677	-0.9364	0.98374	0.71091	-0.6022	0.40248	-1.7372	-2.0276	-0.2693	0.144	0.40249	-0.0485	-1.3719	0.39081	0.19996	0.01637	-0.0146	34.09	0
18	1.16662	0.50212	-0.0673	2.26157	0.4288	0.08947	0.24115	0.13808	-0.9892	0.92217	0.74479	-0.5314	-2.1053	1.12687	0.00308	0.42442	-0.4545	-0.0989	-0.8166	-0.3072	0.0187	-0.062	-0.1039	-0.3704	0.6032	0.10856	-0.0405	-0.0114	2.28	0
18	0.24749	0.27767	1.18547	-0.0926	-1.3144	-0.1501	-0.9464	-1.6179	1.54407	-0.8299	-0.5832	0.52493	-0.4534	0.08139	1.5552	-1.3969	0.78313	0.43662	2.17781	-0.231	1.65018	0.20045	-0.1854	0.42307	0.82059	-0.2276	0.33663	0.25048	22.75	0
22	-1.9465	-0.0449	-0.4056	-1.0131	2.94197	2.95505	-0.0631	0.85555	0.04997	0.57374	-0.0813	-0.2157	0.04416	0.0339	1.19072	0.57884	-0.9757	0.04406	0.4886	-0.2167	-0.5795	-0.7992	0.8703	0.98342	0.3212	0.14965	0.70752	0.0146	0.89	0
22	-2.0743	-0.1215	1.32202	0.41001	0.2952	-0.9595	0.54399	-0.1046	0.47566	0.14945	-0.8566	-0.1805	-0.6552	-0.2798	-0.2117	-0.3333	0.01075	-0.4885	0.50575	-0.3867	-0.4036	-0.2274	0.74243	0.39853	0.24921	0.2744	0.35997	0.24323	26.43	0
23	1.17328	0.3535	0.28341	1.13356	-0.1726	-0.9161	0.36902	-0.3273	-0.2467	-0.0461	-0.1434	0.97935	1.49229	0.10142	0.76148	-0.0146	-0.5116	-0.3251	-0.3909	0.02788	0.067	0.22781	-0.1505	0.43505	0.72482	-0.3371	0.01637	0.03004	41.88	0
23	1.32271	-0.174	0.43456	0.57604	-0.8368	-0.8311	-0.2649	-0.221	-1.0714	0.86856	-0.6415	-0.1113	0.36149	0.17195	0.78217	-1.3559	-0.2169	1.27177	-1.2406	-0.523	-0.2844	-0.3234	-0.0377	0.34715	0.55964	-0.2802	0.04234	0.02882	16	0
23	-0.4143	0.90544	1.72745	1.47347	0.00744	-0.2003	0.74023	-0.0292	-0.5934	-0.3462	-0.0121	0.7868	0.63595	-0.0863	0.0768	-1.4059	0.77559	-0.9429	0.54397	0.09731	0.07724	0.45733	-0.0385	0.64252	-0.1839	-0.2775	0.18269	0.15266	33	0
23	1.05939	-0.1753	1.26613	1.18611	-0.786	0.57844	-0.7671	0.40105	0.6995	-0.0647	1.04829	1.00562	-0.542	-0.0399	-0.2187	0.00448	-0.1936	0.04239	-0.2778	-0.178	0.01368	0.21373	0.01446	0.00295	0.29464	-0.3951	0.08146	0.02422	12.99	0
24	1.23743	0.06104	0.38053	0.76156	-0.3598	-0.4941	0.00649	-0.1339	0.43881	-0.2074	-0.9292	0.52711	0.34868	-0.1525	-0.2184	-0.1916	-0.1166	-0.6338	0.34842	-0.0664	-0.2457	-0.5309	-0.0443	0.07917	0.50914	0.28886	-0.0227	0.01184	17.28	0
25	1.11401	0.08555	0.4937	1.33576	-0.3002	-0.0108	-0.1188	0.18862	0.20569	0.08226	1.13356	0.6267	-1.4928	0.52079	-0.6746	-0.5291	0.15826	-0.3988	-0.1457	-0.2738	-0.0532	-0.0048	-0.0315	0.19805	0.56501	-0.3377	0.02906	0.00445	4.45	0
26	-0.5299	0.87389	1.34725	0.14546	0.41421	0.10022	0.71121	0.17607	-0.2867	-0.4847	0.87249	0.85164	-0.5717	0.10097	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.04695	0.2081	-0.1855	0.00103	0.09882	-0.5529	-0.0733	0.02331	6.14	0
26	-0.5299	0.87389	1.34725	0.14546	0.41421	0.10022	0.71121	0.17607	-0.2867	-0.4847	0.87249	0.85164	-0.5717	0.10097	-1.5198	-0.2844	-0.3105	-0.4042	-0.8234	-0.2903	0.04695	0.2081	-0.1855	0.00103	0.09882	-0.5529	-0.0733	0.02331	6.14	0
26	-0.5354	0.86527	1.35108	0.14758	0.43368	0.08698	0.69304	0.17974	-0.2856	-0.4825	0.8718	0.85345	-0.5718	0.10225	-1.52	-0.2859	-0.3096	-0.4039	-0.8237	-0.2833	0.04953	0.20654	-0.1871	0.00075	0.09812	-0.5535	-0.0783	0.02543	1.77	0
26	-0.5354	0.86527	1.35108	0.14758	0.43368	0.08698	0.69304	0.17974	-0.2856	-0.4825	0.8718	0.85345	-0.5718	0.10225	-1.52	-0.2859	-0.3096	-0.4039	-0.8237	-0.2833	0.04953	0.20654	-0.1871	0.00075	0.09812	-0.5535	-0.0783	0.02543	1.77	0
27	-0.246	0.47327	1.69574	0.26241	-0.0109	-0.6108	0.79394	-0.2473	0.13888	-0.401	-0.8121	-0.1835	-0.6301	-0.2862	-0.337	-0.4281	-0.0306	-0.5046	0.35518	0.04572	-0.1946	-0.3351	-0.0782	0.39278	-0.0312	0.19804	-0.1753	-0.2036	30.49	0
27	-1.4522	1.76512	0.61167	1.17682	-0.446	0.24683	-0.2576	1.09247	-0.6075	0.04716	0.78373	1.09639	-0.2681	0.76865	-0.5244	-0.8088	0.71039	-0.1184	0.97929	0.00871	0.08228	0.32578	-0.0691	0.02096	-0.0447	-0.2434	0.14918	0.12056	1.8	0
29	0.99637	-0.1226	0.54682	0.70658	0.13456	1.157	-0.2946	0.40743	0.33786	-0.4082	0.68265	1.25372	0.39192	-0.0375	0.7953	-1.6682	1.32299	-2.83	-1.4634	-0.2035	-0.0763	0.10887	0.16223	-0.5756	0.10979	0.37381	0.05055	0.00511	20.53	0
29	1.11088	0.16872	0.51714	1.32541	-0.1916	0.0195	-0.0318	0.11762	0.01766	0.04486	1.34507	1.28634	-0.2523	0.27446	-0.8104	-0.587	0.08745	-0.5505	-0.1547	-0.1901	-0.0377	0.0957	-0.0482	0.23211	0.6062	-0.3421	0.03677	0.00748	6.54	0
32	1.24905	-0.6247	-0.7106	-0.9916	1.42997	3.69298	-1.0902	0.96729	0.85015	-0.3071	-0.4562	0.22998	-0.0169	-0.2208	0.36242	0.31522	-0.5123	0.11899	0.57472	0.09785	-0.0063	0.0092	-0.1295	1.11297	0.50038	1.19655	-0.0482	0.00509	29.89	0
32	-2.0089	2.19853	0.14424	1.15943	-0.8152	0.18229	-0.6171	1.53082	-0.5868	0.12988	0.65848	1.26505	-0.2065	1.0136	-0.5494	-0.708	0.86964	-0.0957	0.9444	0.02874	0.09492	0.29498	0.01108	0.01525	0.03421	-0.2361	0.12829	0.11799	2.35	0
33	-0.6079	1.03135	1.74045	1.23211	0.41859	0.11917	0.85089	-0.1763	-0.2435	0.14846	-0.387	0.3983	0.48192	-0.3654	0.23554	-1.3478	0.50465	-0.7984	0.75971	0.25432	-0.0873	0.25832	-0.2648	0.11828	0.17351	-0.217	0.09431	-0.033	14.8	0
33	-0.9357	0.17042	2.74626	-1.078	-0.3056	0.01158	-0.2962	0.40278	-0.0405	-0.852	1.55211	0.95356	0.41543	-0.2339	0.45234	0.84437	-0.8227	0.33882	-1.2142	-0.0054	0.40121	1.06486	-0.1583	0.29551	-0.2594	0.75419	0.04666	0.09395	9.1	0

Figure 6.1 Credit Card Dataset

Each method used in this model takes input from the same credit card dataset. Instead of the names, 28 variables out of the total 30 are named from V1 to V28 and the last two variables are named as amount and class. Along with the credit card dataset, spiral and grid are two other datasets which are used for Artificial Neural Network(ANN) method. The value used in class variable is either 0 or 1. If the financial statement is fraud then the predicted value should be set to 1 and if the financial statement is not fraud then the predicted value is set to 0.

6.2 RESULT ANALYSIS

The credit card dataset is given as input for all the four methods separately and the accuracy of each method will be compared with the accuracy of other methods of the fraudulent financial statement detection model. When the attribute named class has the predicted value as 1 then the financial statement is considered as a fraudulent financial statement and if the value is predicted to be 0 then financial statement is fraud free.

Random Forest algorithm splits the dataset into two parts for processing i.e., 70% as training data and remaining 30% as test data. Since the Random Forest method deals with fully grown decision trees, the accuracy of this method is lower than other methods. Random Forest Algorithm has low bias value and higher variance value.

Artificial Neural Network method splits the dataset into the following manner i.e., 60% as training data, 20% as validation data and remaining 20% as test data. The implementation of this method gives us the maximum accuracy of 87%. Gradient Boosting Machine Algorithm is considered as a better alternative algorithm for random forest because Gradient Boosting Machine constructs a new tree with the help of previously constructed tree and also this algorithm has high bias and low variance but the Random Forest Algorithm has low bias and high variance. The values predicted in the Gradient Boosting Machine Algorithm are mean square error , root mean square error, mean absolute error and root mean squared logarithmic error.

Generalized Linear Model is mainly used in statistical analysis and the data is split into three such as 60% as training data, 20% as validation data and remaining 20% as test data. The multinomial model will be constructed initially with the default parameters with the help of Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm. The default lambda value is 0.001. In order to update the Generalized Linear Model Estimator the lambda value is lowered to 0.0001 which leads to improvement in the mean square error value, mean absolute error value and then the confusion matrix is built. Two classes are used in this model which implies fraud and not fraud respectively. A binomial classifier is built and features are added to the binomial model to improve its predictive capacity and then the final multinomial classifier with a lambda value of 0.001 is constructed. The hit ratio gradually increases since the first multinomial model.

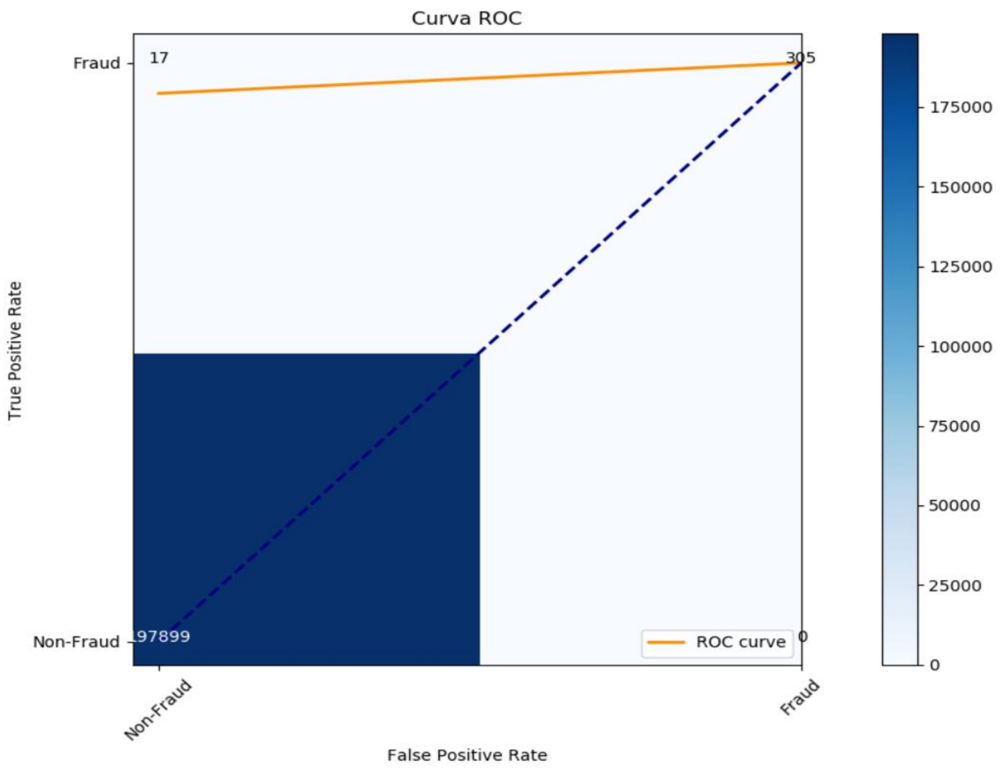


Figure 6.2 Training Data Result

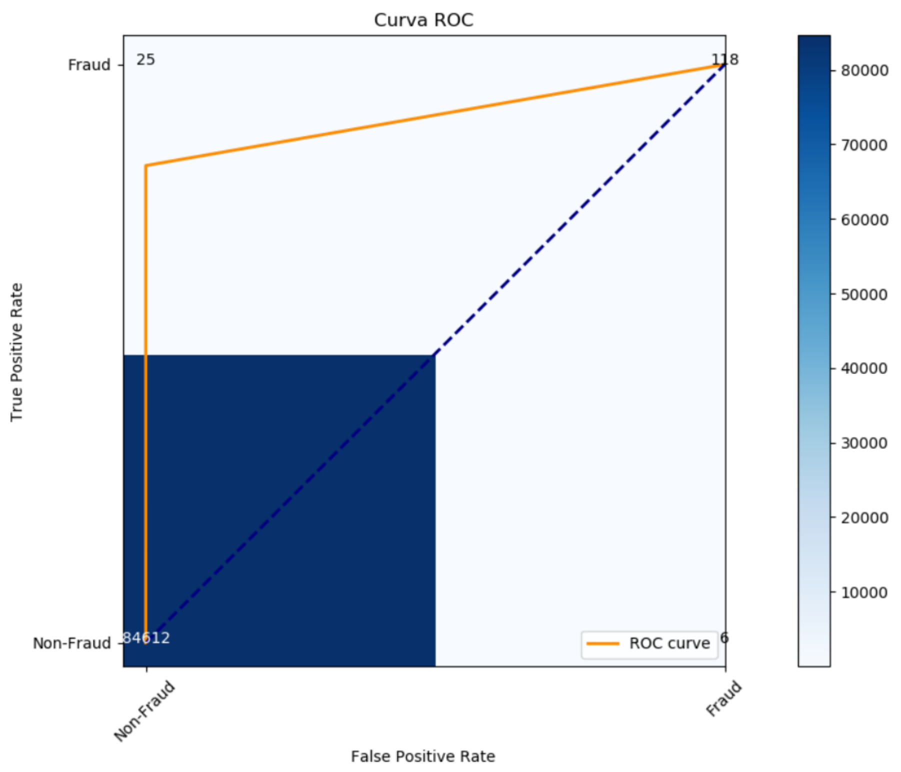


Figure 6.3 Test Data Result

Figure 6.2 and Figure 6.3 represents the histogram representation of training data and test data respectively. With the help of the true positive rate and false positive rate calculated in the confusion matrix the output graph which consists of receiver operating characteristic curve is produced.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

A company’s financial statement is the key basis for all investor judgments, and is the last line of defense for investor interests. If management attempts to withhold information, even when independent CPAs, investment banking and securities analysts are involved, investors can experience significant losses. Based on the empirical results of this study, the accuracy of the DT CHAID, combined with CART, in detecting fraudulent financial statements, is relatively high. It can therefore be used as a tool to help auditors in the detection of fraudulent financial statements. The research findings can provide a reference for investors, shareholders, company managers, credit rating institutions, auditors, CPAs (certified public accountants), securities analysts, financial regulatory authorities, and relevant academic institutions.

7.2 FUTURE ENHANCEMENT

In our future scope we planned to detect the fraudulent financial statements of any organization using artificial intelligence concept which prevents the financial loss incurred by the organization and improves the accuracy. The project can be extended in such a way that methods with higher accuracy will be used to detect the fraudulent financial statements effectively which helps to detect the fraudulent behavior prior to its occurrence.

APPENDIX I - CODING

RANDOM FOREST:

```
import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

import seaborn as sns

import os

import h2o

import numpy as np

data = reviews = pd.read_csv('dataset.csv')

data['Class'] = data['Class'].astype('category')

cols = list(data.columns)

data.groupby('Class')['Class'].count()

h2o.init()

creditcard = h2o.upload_file(path ='dataset.csv')

creditcard['Class'] = creditcard['Class'].asfactor()

df = creditcard.as_data_frame()

cols = list(df.columns)

df.isnull().sum()

df.query('Amount == 0').groupby('Class').count()

df.query('Amount != 0').to_csv('creditcard_amount_positive.csv')

creditcard = h2o.upload_file(path ='creditcard_amount_positive.csv')

creditcard['Class'] = creditcard['Class'].asfactor()

df = creditcard.as_data_frame()

from scipy import stats

variables = cols[:-1]
```

```
gs = gridspec.GridSpec(num_cols, 1)
for i, cn in enumerate(df[cols[:-1]]):
    ax = plt.subplot(gs[i])
    sns.distplot(df[cn][df.Class == 1], bins=50)
    sns.distplot(df[cn][df.Class == 0], bins=50)
    ax.set_xlabel('')
    ax.set_title('histogram of feature: ' + str(cn))
plt.show()
from scipy import stats
variables = cols[:-1]
keep = []
p_value_alpha = 0.05 #defult p-value for statistical significance
for variable in variables:
    fraud_v = df[variable][df.Class == 1]
    not_fraud_v = df[variable][df.Class == 0].sample(len(fraud_v))
    p_value = stats.ttest_ind(not_fraud_v, fraud_v).pvalue
    if p_value >= p_value_alpha:
        print("Distributions are equal. Discard {} variable".format(variable))
    else:
        print("Distributions are diferent. Keep {} variable".format(variable))
        keep.append(variable)
from h2o.estimators.random_forest import H2ORandomForestEstimator
train, valid = creditcard.split_frame(ratios=[0.7])
response_var = 'Class'
features = [col for col in cols if col != response_var]
naive_rf_model = H2ORandomForestEstimator()
```

```
performance_train = naive_rf_model.model_performance(train=True)
import itertools
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
       print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')

    plt.xlabel('Predicted label')

preds = naive_rf_model.predict(train)

cm = confusion_matrix(train.as_data_frame()['Class'], preds.as_data_frame()['predict'])

plot_confusion_matrix(cm, ['Non-Fraud', 'Fraud'], False)

fpr, tpr, threshold = roc_curve(train.as_data_frame()['Class'],
preds.as_data_frame()['predict'])

plt.plot(fpr, tpr, color='darkorange',

        lw=2, label='ROC curve')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([-0.04, 1.0])

plt.ylim([-0.04, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Curva ROC')

plt.legend(loc="lower right")

plt.show()

predictions = naive_rf_model.predict(valid)

cm = confusion_matrix(valid.as_data_frame()['Class'], predictions.as_data_frame()['predict'])

plot_confusion_matrix(cm, ['Non-Fraud', 'Fraud'], False)

fpr, tpr, threshold = roc_curve(valid.as_data_frame()['Class'],
predictions.as_data_frame()['predict'])

plt.plot(fpr, tpr, color='darkorange',

        lw=2, label='ROC curve')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([-0.04, 1.0])

plt.ylim([-0.04, 1.05])

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

plt.title('Curva ROC')

plt.legend(loc="lower right")

plt.show()
```

DEEP LEARNING:

```
import h2o

h2o.init(max_mem_size = 2)            #uses all cores by default

h2o.remove_all()                       #clean slate, in case cluster was already running

from h2o.estimators.deeplearning import H2OAutoEncoderEstimator,
H2ODeepLearningEstimator

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from h2o.estimators.deeplearning import H2OAutoEncoderEstimator,
H2ODeepLearningEstimator

from h2o.estimators.gbm import H2OGradientBoostingEstimator

import os

spiral = h2o.import_file(path = os.path.realpath("input/spiral.csv"))

grid  = h2o.import_file(path = os.path.realpath("input/grid.csv"))

spiral_df = spiral.as_data_frame(use_pandas=True)

grid_df = grid.as_data_frame(use_pandas=True)

grid_x, grid_y = grid_df.x.reshape(201,201), grid_df.y.reshape(201,201)

spiral_r = spiral_df[spiral_df.color == "Red"]

spiral_k = spiral_df[spiral_df.color == "Black"]

spiral_xr, spiral_yr = spiral_r[spiral_r.columns[0]], spiral_r[spiral_r.columns[1]]

spiral_xk, spiral_yk = spiral_k[spiral_k.columns[0]], spiral_k[spiral_k.columns[1]]
```

```
markersize_ = 7**2

plt.figure(figsize = (5,5))

plt.scatter(spiral_xr, spiral_yr, c = 'r', s=markersize_)

plt.scatter(spiral_xk, spiral_yk, c = 'k', s=markersize_)

plt.axis([-1.5, 1.5, -1.5, 1.5])

plt.title("Spiral");

X = spiral.col_names[0:2]

y = spiral.col_names[2]

dl_model = H2ODeepLearningEstimator(epochs=1000)

dl_model.train(X, y, spiral)

gbm_model = H2OGradientBoostingEstimator()

gbm_model.train(X, y, spiral)

drf_model = H2ORandomForestEstimator()

drf_model.train(X, y, spiral)

glm_model = H2OGeneralizedLinearEstimator(family="binomial")

glm_model.fit(spiral[X], spiral[y])                    #model.fit() example

models = [dl_model, gbm_model, drf_model, glm_model]

m_names = ["Deep Learning", "Gradient Boosted Method", "Distributed Random Forest",
"Generalized Linear Model"]

def plot_spirals(models, model_names):

    fig, ax = plt.subplots(2,2, figsize=(12,12))

    for k, subplot in enumerate(ax.flatten()):

        subplot.scatter(spiral_xr, spiral_yr, c = 'r', s=markersize_)

        subplot.scatter(spiral_xk, spiral_yk, c = 'k', s=markersize_)

        subplot.axis([-1.5, 1.5, -1.5, 1.5])

        subplot.set_title(model_names[k])

        subplot.set_ylabel('y')

        pred_z = models[k].predict(grid).as_data_frame(True)
    subplot.contour(grid_x, grid_y, (pred_z['predict'] ==
'Black').astype(np.int).reshape(201,201), colors='b')

dl_1 = H2ODeepLearningEstimator(epochs=1)

dl_1.train(X, y, spiral)

dl_250 = H2ODeepLearningEstimator(checkpoint=dl_1, epochs=250)

dl_250.train(X, y, spiral)

dl_500 = H2ODeepLearningEstimator(checkpoint=dl_250, epochs=500)

dl_500.train(X, y, spiral)

dl_750 = H2ODeepLearningEstimator(checkpoint=dl_500, epochs=750)

dl_750.train(X, y, spiral)

models_dl = [dl_1, dl_250, dl_500, dl_750]

m_names_dl = ["DL " + str(int(model.get_params()['epochs']['actual_value'])) + " Epochs"
for model in models_dl]

plot_spirals(models_dl, m_names_dl)

dl_1 = H2ODeepLearningEstimator(hidden=[1000], epochs=500)

dl_1.train(X, y, spiral)

dl_2 = H2ODeepLearningEstimator(hidden=[200,200], epochs=500)

dl_2.train(X, y, spiral)

dl_3 = H2ODeepLearningEstimator(hidden=[42,42,42], epochs=500)

dl_3.train(X, y, spiral)

dl_4 = H2ODeepLearningEstimator(hidden=[11,13,17,19], epochs = 1000)

dl_4.train(X, y, spiral)

# It is clear that different configurations can achieve similar performance, and that tuning will
be required for optimal performance.

models_network = [dl_1, dl_2, dl_3, dl_4]

m_names_network = ["1000", "200 x 200", "42 x 42 x 42", "11 x 13 x 17 x 19"]

plot_spirals(models_network, m_names_network)

models_act = []

m_names_act = []
```

```
for i,method in enumerate(["Tanh","Maxout","Rectifier","RectifierWithDropout"]):
    models_act.append(H2ODeepLearningEstimator(activation=method, hidden=[100,100], epochs=1000))
    models_act[i].train(X, y, spiral)
    m_names_act.append("DL "+ method + " Activation")
plot_spirals(models_act, m_names_act)
creditcard_df = h2o.import_file(path = os.path.realpath("datast.csv"))
train, valid, test = creditcard_df.split_frame([0.6, 0.2], seed=1234)
creditcard_X = creditcard_df.col_names[:-1]     #last column is cover_type,
creditcard_y = creditcard_df.col_names[-1]
creditcard_model_v1 = H2ODeepLearningEstimator(model_id="creditcard_v1", epochs=1, variable_importances=True)
creditcard_model_v1.train(creditcard_X, creditcard_y, training_frame = train, validation_frame = valid)
print creditcard_model_v1
var_df = pd.DataFrame(creditcard_model_v1.varimp(),
             columns=["Variable", "Relative Importance", "Scaled Importance", "Percentage"])
print var_df.shape
var_df.head(10)
creditcard_model_v2 = H2ODeepLearningEstimator(
    model_id="creditcard_v2",
    hidden=[32,32,32],                  ## small network, runs faster
    epochs=1000000,                      ## hopefully converges earlier...
    score_validation_samples=10000,      ## sample the validation dataset (faster)
    stopping_rounds=2,
    stopping_metric="MSE", ## alternatives: "MSE","logloss","r2"
    stopping_tolerance=0.01)
creditcard_model_v2.train(creditcard_X, creditcard_y, training_frame=train, validation_frame=valid)
```

```
cov_v2_df = creditcard_model_v2.score_history()

cov_v2_df

plt.plot(cov_v2_df['training_classification_error'], label="training_classification_error")

plt.plot(cov_v2_df['validation_classification_error'], label="validation_classification_error")

plt.title("Covertype Deep Learner (Early Stop)")

plt.legend();

# ### Adaptive Learning Rate

creditcard_model_tuned = H2ODeepLearningEstimator(

  model_id="creditcard_tuned",

  overwrite_with_best_model=False,

  hidden=[128,128,128],            # more hidden layers -> more complex interactions

  epochs=10,                       # to keep it short enough

  score_validation_samples=10000,  # downsample validation set for faster scoring

  score_duty_cycle=0.025,          # don't score more than 2.5% of the wall time

  adaptive_rate=False,             # manually tuned learning rate

  rate=0.01,

  rate_annealing=0.000002,

  momentum_start=0.2,              # manually tuned momentum

  momentum_stable=0.4,

  momentum_ramp=10000000,

  l1=0.00001,                      # add some L1/L2 regularization

  l2=0.00001,

  max_w2=10                        # helps stability for Rectifier

)

creditcard_model_tuned.train(creditcard_X, creditcard_y, training_frame=train,
validation_frame=valid)

cov_tuned_df = creditcard_model_tuned.scoring_history()

cov_tuned_df
```

```
plt.plot(cov_tuned_df['training_classification_error'], label="training_classification_error")

plt.plot(cov_tuned_df['validation_classification_error'],
label="validation_classification_error")

plt.title("Covertype Deep Learner (Tuned)")

plt.legend();

pred = creditcard_model_tuned.predict(test[0:-1]).as_data_frame(use_pandas=True)

test_actual = test.as_data_frame(use_pandas=True)['Cover_Type']

(test_actual == pred['predict']).mean()
```

GRADIENT BOOSTING METHOD:

```
import h2o

import os

h2o.init(max_mem_size = "2G")             #specify max number of bytes. uses all cores by
default.

h2o.remove_all() #clean slate, in case cluster was already running

from h2o.estimators.gbm import H2OGradientBoostingEstimator

creditcard_df = h2o.import_file(os.path.realpath("dataset.csv"))

# 60% for training

# 20% for validation (hyper parameter tuning)

# 20% for final testing

#split the data as described above

train, valid, test = creditcard_df.split_frame([0.6, 0.2], seed=1234)

#Prepare predictors and response columns

creditcard_X = creditcard_df.col_names[:-1]     #last column is Class, our desired response
variable

creditcard_y = creditcard_df.col_names[-1]

gbm_v1 = H2OGradientBoostingEstimator(

    model_id="gbm_creditcard_v1",
gbm_v1.train(creditcard_X, creditcard_y, training_frame=train, validation_frame=valid)
gbm_v1.score_history()
gbm_v1.hit_ratio_table(valid, train = FALSE, valid = FALSE, xval = FALSE)
# ### GBM Round 2
gbm_v2 = H2OGradientBoostingEstimator(
    ntrees=20,
    learn_rate=0.2,
    max_depth=10,
    stopping_tolerance=0.01, #10-fold increase in threshold as defined in rf_v1
    stopping_rounds=2,
    score_each_iteration=True,
    model_id="gbm_creditcard_v2",
    seed=2000000
gbm_v2.train(creditcard_X, creditcard_y, training_frame=train, validation_frame=valid)
gbm_v2.hit_ratio_table(valid=True)
# ### GBM: Third Time
gbm_v3 = H2OGradientBoostingEstimator(
    ntrees=30,
    learn_rate=0.3,
    max_depth=10,
    sample_rate=0.7,
    col_sample_rate=0.7,
    stopping_rounds=2,
    stopping_tolerance=0.01, #10-fold increase in threshold as defined in rf_v1
    score_each_iteration=True,
    model_id="gbm_creditcard_v3",
    seed=2000000
```

```
gbm_v3.train(creditcard_X, creditcard_y, training_frame=train, validation_frame=valid)

gbm_v3.hit_ratio_table(valid=True)
```

GENERALIZED LINEAR METHOD

```
import h2o

h2o.init(max_mem_size = "2G")           #specify max number of bytes. uses all cores by
default.

h2o.remove_all()                        #clean slate, in case cluster was already running

from h2o.estimators.glm import H2OGeneralizedLinearEstimator

import pandas as pd

import numpy as np

creditcard_df = h2o.import_file(os.path.realpath("dataset.csv"))

train, valid, test = creditcard_df.split_frame([0.7, 0.15], seed=1234)

creditcard_X = creditcard_df.col_names[:-1]     #last column is Cover_Type, our desired
response variable

creditcard_y = creditcard_df.col_names[-1]

glm_multi_v1 = H2OGeneralizedLinearEstimator(

                model_id='glm_v1',          #allows us to easily locate this model in Flow

                family='multinomial',

                solver='L_BFGS')

glm_multi_v1.train(creditcard_X, creditcard_y, training_frame=train,
validation_frame=valid)

glm_multi_v1

glm_multi_v1.hit_ratio_table(valid=True)

glm_multi_v2 = H2OGeneralizedLinearEstimator(

                model_id='glm_v2',

                family='multinomial',

                solver='L_BFGS',
          Lambda=0.0001           #default value 0.001
)
glm_multi_v2.train(creditcard_X, creditcard_y, training_frame=train,
validation_frame=valid)
glm_multi_v2
glm_multi_v2.hit_ratio_table(valid=True)
glm_multi_v2.confusion_matrix(valid)
c1 = creditcard_df[creditcard_df['Cover_Type'] == 'class_1']
c2 = creditcard_df[creditcard_df['Cover_Type'] == 'class_2']
df_b = c1.rbind(c2)
train_b, valid_b, test_b = df_b.split_frame([0.7, 0.15], seed=1234)
glm_binom_v1 = H2OGeneralizedLinearEstimator(
          model_id="glm_v3",
          solver="L_BFGS",
          family="binomial")
glm_binom_v1.train(creditcard_X, creditcard_y, training_frame=train_b,
validation_frame=valid_b)
glm_binom_v1.accuracy()
def cut_column(train_df, train, valid, test, col):
  only_col= train_df[col]               #Isolate the column in question from the training
frame
  counts, breaks = np.histogram(only_col, bins=20)   #Generate counts and breaks for our
histogram
  min_val = min(only_col)-1               #Establish min and max values
  max_val = max(only_col)+1
  new_b = [min_val]                  #Redefine breaks such that each bucket has
enough support
  for i in xrange(19):
    if counts[i] > 1000 and counts[i+1] > 1000:
          new_b.append(breaks[i+1])
  new_b.append(max_val)
  names = [col + '_' + str(x) for x in xrange(len(new_b)-1)]  #Generate names for buckets, 
these will be categorical names
  train[col+"_cut"] = train[col].cut(breaks=new_b, labels=names)
  valid[col+"_cut"] = valid[col].cut(breaks=new_b, labels=names)
  test[col+"_cut"] = test[col].cut(breaks=new_b, labels=names)
def add_features(train, valid, test):
  train_df = train.as_data_frame(True)
  cut_column(train_df, train, valid, test, "Elevation")
  cut_column(train_df, train, valid, test, "Hillshade_Noon")
  cut_column(train_df, train, valid, test, "Hillshade_9am")
  cut_column(train_df, train, valid, test, "Hillshade_3pm")
  cut_column(train_dfs, train, valid, test, "Horizontal_Distance_To_Hydrology")
  cut_column(train_df, train, valid, test, "Slope")
  cut_column(train_df, train, valid, test, "Horizontal_Distance_To_Roadways")
  cut_column(train_df, train, valid, test, "Aspect")
  interaction_cols1 = ["Elevation_cut",
                       "Wilderness_Area",
                       "Soil_Type",
                       "Hillshade_Noon_cut",
                       "Hillshade_9am_cut",
                       "Hillshade_3pm_cut",
                       "Horizontal_Distance_To_Hydrology_cut",
                       "Slope_cut",
                       "Horizontal_Distance_To_Roadways_cut",
                       "Aspect_cut"]
  train_cols = train.interaction(factors=interaction_cols1,    #Generate pairwise columns
                        pairwise=True,
                        max_factors=1000,
                        min_occurrence=100,
                        destination_frame="itrain")
valid_cols = valid.interaction(factors=interaction_cols1,
                        pairwise=True,
                        max_factors=1000,
                        min_occurrence=100,
                        destination_frame="ivalid")
test_cols = test.interaction(factors=interaction_cols1,
                        pairwise=True,
                        max_factors=1000,
                        min_occurrence=100,
                        destination_frame="itest")
train = train.cbind(train_cols)                    #Append pairwise columns to H2OFrames
valid = valid.cbind(valid_cols)
test = test.cbind(test_cols)
#Add a three-way interaction for Hillshade
interaction_cols2 = ["Hillshade_Noon_cut","Hillshade_9am_cut","Hillshade_3pm_cut"]
train_cols = train.interaction(factors=interaction_cols2,    #Generate pairwise columns
                        pairwise=False,
                        max_factors=1000,
                        min_occurrence=100,
                        destination_frame="itrain")
valid_cols = valid.interaction(factors=interaction_cols2,
                        max_factors=1000,
                        min_occurrence=100,
                        destination_frame="ivalid")
    test_cols = test.interaction(factors=interaction_cols2,
    train = train.cbind(train_cols)                 #Append pairwise columns to H2OFrames
    valid = valid.cbind(valid_cols)
    test = test.cbind(test_cols)
    return train, valid, test
train_bf, valid_bf, test_bf = add_features(train_b, valid_b, test_b)
glm_binom_feat_1 = H2OGeneralizedLinearEstimator(family='binomial', solver='L_BFGS',
model_id='glm_v4')
glm_binom_feat_1.train(creditcard_X, creditcard_y, training_frame=train_bf,
validation_frame=valid_bf)
glm_binom_feat_1.accuracy(valid=True)
glm_binom_feat_2 = H2OGeneralizedLinearEstimator(family='binomial', solver='L_BFGS',
model_id='glm_v5', Lambda=0.001)
glm_binom_feat_2.train(creditcard_X, creditcard_y, training_frame=train_bf,
validation_frame=valid_bf)
glm_binom_feat_3 = H2OGeneralizedLinearEstimator(family='binomial',
model_id='glm_v6', lambda_search=True)
glm_binom_feat_3.train(creditcard_X, creditcard_y, training_frame=train_bf,
validation_frame=valid_bf)
glm_binom_feat_3.accuracy(valid=True)
train_f, valid_f, test_f = add_features(train, valid, test)
glm_multi_v3 = H2OGeneralizedLinearEstimator(
                solver='L_BFGS',
                Lambda=0.0001)
glm_multi_v3.train(creditcard_X, creditcard_y, training_frame=train_f,
validation_frame=valid_f)
glm_multi_v3.hit_ratio_table(valid=True)
```

APPENDIX 2 - SCREENSHOTS

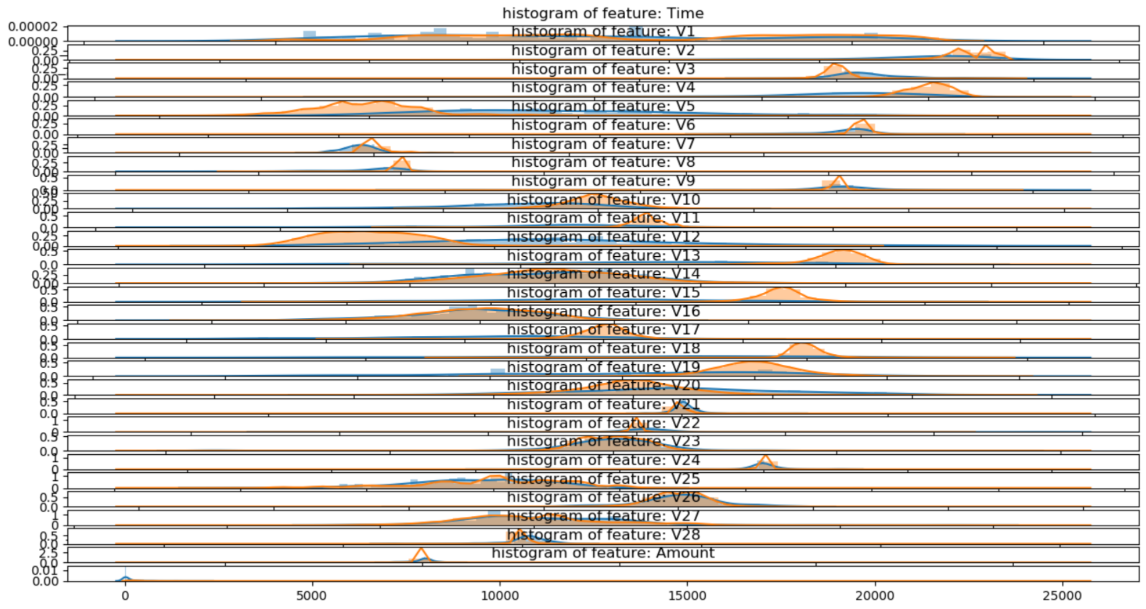
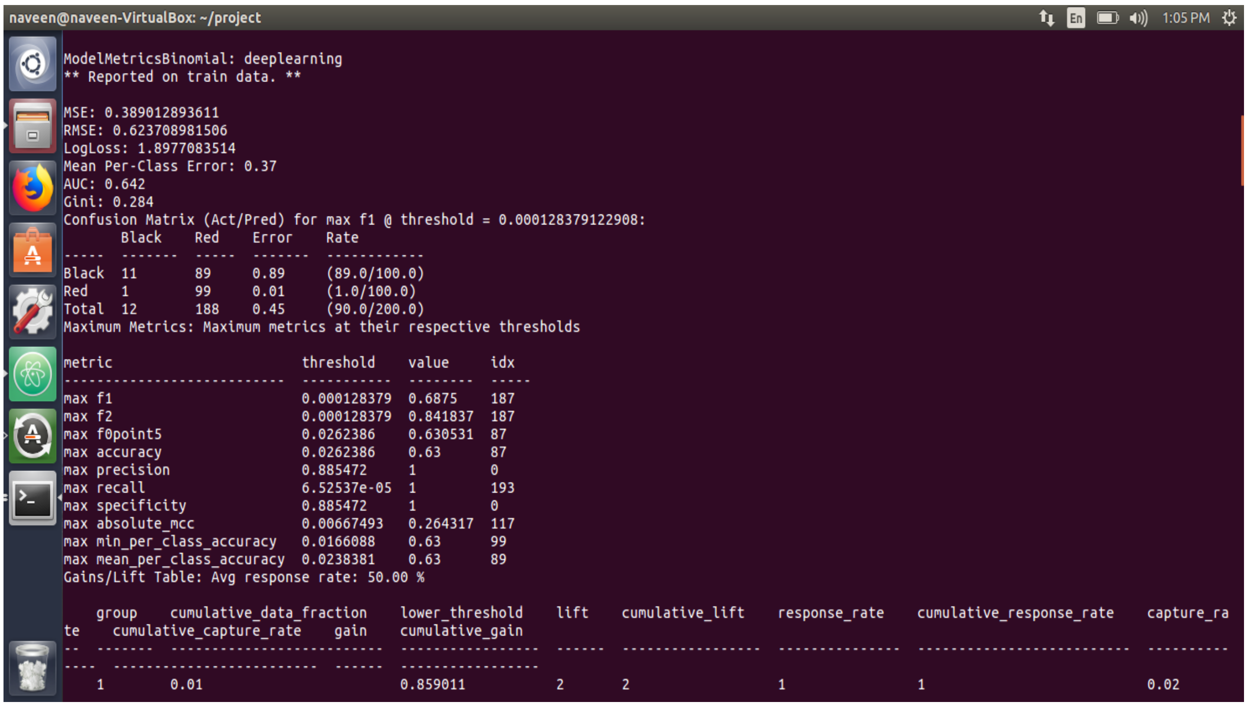


Figure A.2.1 Variable Selection



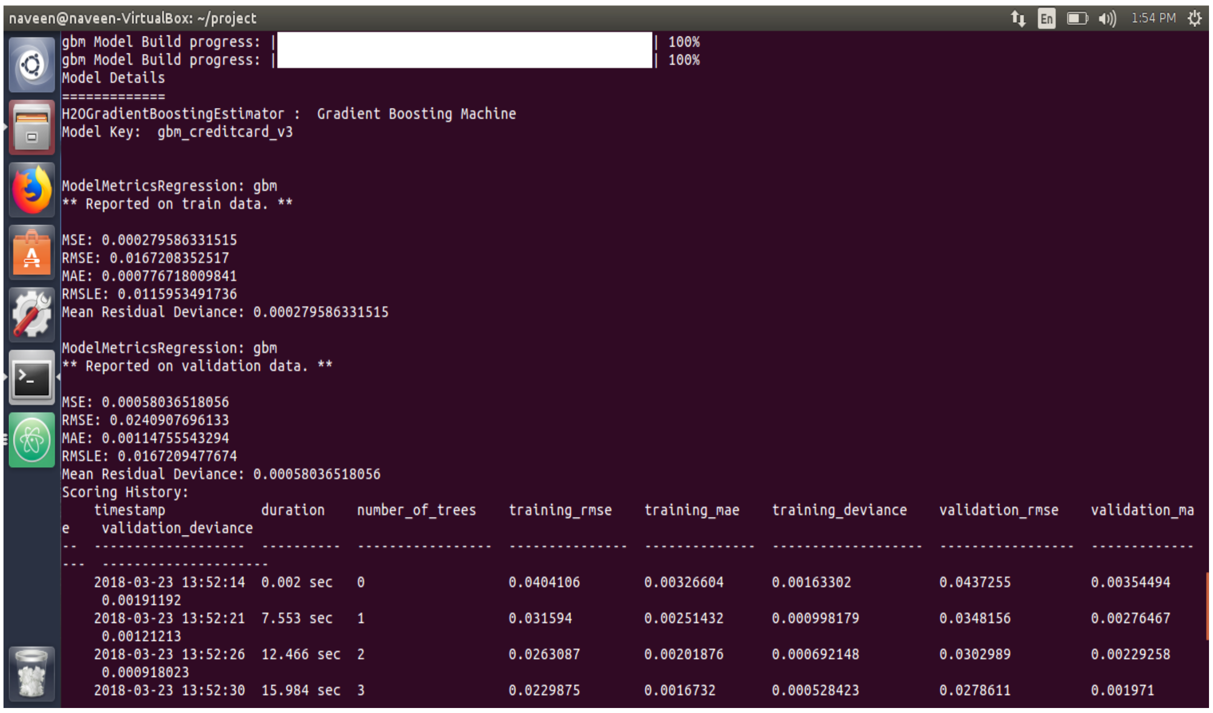
```
ModelMetricsBinomial: deeplearning
** Reported on train data. **

MSE: 0.389012893611
RMSE: 0.623708981506
LogLoss: 1.8977083514
Mean Per-Class Error: 0.37
AUC: 0.642
Gini: 0.284
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.000128379122908:
       Black    Red    Error    Rate
-----  -------  -----  -------  ------------
Black  11       89     0.89     (89.0/100.0)
Red    1        99     0.01     (1.0/100.0)
Total  12       188    0.45     (90.0/200.0)
Maximum Metrics: Maximum metrics at their respective thresholds

metric                       threshold    value     idx
---------------------------  -----------  --------  -----
max f1                       0.000128379  0.6875    187
max f2                       0.000128379  0.841837  187
max f0point5                 0.0262386    0.630531  87
max accuracy                 0.0262386    0.63      87
max precision                0.885472     1         0
max recall                   6.52537e-05  1         193
max specificity              0.885472     1         0
max absolute_mcc             0.00667493   0.264317  117
max min_per_class_accuracy   0.0166088    0.63      99
max mean_per_class_accuracy  0.0238381    0.63      89
Gains/Lift Table: Avg response rate: 50.00 %

    group    cumulative_data_fraction    lower_threshold    lift    cumulative_lift    response_rate    cumulative_response_rate    capture_ra
te    cumulative_capture_rate    gain    cumulative_gain
--  -------  -------------------------  -----------------  ------  -----------------  ---------------  --------------------------  ----------
----  ------------------------  ------  -----------------
    1        0.01                        0.859011           2       2                  1                1                           0.02
```

Figure A.2.2 Accuracy of ANN



```
naveen@naveen-VirtualBox: ~/project
gbm Model Build progress: |                                               | 100%
gbm Model Build progress: |                                               | 100%
Model Details
=============
H2OGradientBoostingEstimator :  Gradient Boosting Machine
Model Key:  gbm_creditcard_v3


ModelMetricsRegression: gbm
** Reported on train data. **

MSE: 0.000279586331515
RMSE: 0.0167208352517
MAE: 0.000776718009841
RMSLE: 0.0115953491736
Mean Residual Deviance: 0.000279586331515

ModelMetricsRegression: gbm
** Reported on validation data. **

MSE: 0.00058036518056
RMSE: 0.0240907696133
MAE: 0.00114755543294
RMSLE: 0.0167209477674
Mean Residual Deviance: 0.00058036518056
Scoring History:
    timestamp            duration    number_of_trees    training_rmse    training_mae    training_deviance    validation_rmse    validation_ma
e    validation_deviance
--  -------------------  ----------  -----------------  ---------------  --------------  -------------------  -----------------  -------------
---  --------------------
    2018-03-23 13:52:14  0.002 sec   0                  0.0404106        0.00326604      0.00163302           0.0437255          0.00354494
     0.00191192
    2018-03-23 13:52:21  7.553 sec   1                  0.031594         0.00251432      0.000998179          0.0348156          0.00276467
     0.00121213
    2018-03-23 13:52:26  12.466 sec  2                  0.0263087        0.00201876      0.000692148          0.0302989          0.00229258
     0.000918023
    2018-03-23 13:52:30  15.984 sec  3                  0.0229875        0.0016732       0.000528423          0.0278611          0.001971
```

Figure A.2.3 GBM Result

REFERENCES

1. Chen S (2016), ‘Detection of fraudulent financial statements using the hybrid data mining approach’, SpringerPlus, Vol.5, No.1, pp. 89.

2. Kamarudin K A et al., (2012), ‘Aggressive financial reporting and corporate fraud’, Procedia-Social and Behavioral Sciences,Vol.65, pp.638-643.

3. Ravisankar P et al., (2011), ’Detection of financial statement fraud and feature selection using data mining techniques’, Decision Support Systems, Vol.50,No.2, pp. 491-500.

4. Yeh C C et al., (2010), ‘A hybrid approach of DEA, rough set and support vector machines for business failure prediction’, Expert Systems with Applications, Vol.37,No.2, pp.1535-1541.

5. Zhou W et al., (2011), ’Detecting evolutionary financial statement fraud’, Decision Support Systems, Vol.50,No.3, pp.570-575.