UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

TOLL FRAUD DETECTION IN VOIP NETWORKS USING ARTIFICIAL NEURAL NETWORKS

NICHOLAS KOISER

P58/61547/2010

SUPERVISOR:

DR. ROBERT OBOKO.

Report submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the School of Computing and Informatics, University of Nairobi.

Declaration

I declare that this research project proposal as my original work and has not been submitted to the University of Nairobi and any other university to the best of my knowledge for the same purpose in the same scope and area of research case study.

NAME: NICHOLAS KOISER   REG. NO: P58/61547/2010

SIGNATURE…………………………   DATE…………………………

This research project report is submitted for presentation with my approval as the

University Research project supervisor

NAME: DR.ROBERT OBOKO

SIGNATURE………………………………… DATE…………………………………..

Dedication

I dedicate this work to the memory of my late father and mother who inspired, encouraged and provided for my education.

This dedication also extends to my dear wife Christine and daughters Elsie and Ellah for their support during my project work.

Acknowledgement

I thank the almighty God for His blessings upon my life, His providential care all through the time of my study and for His strength that has kept me going against all odds.

I would also like to thank my project supervisor, Dr. Robert Oboko for his guidance and dedication of time towards the achievement of this project's objectives. I wish to also extend this gratitude to the panel members who also gave much needed advice throughout the project.

I also thank my class mates for the encouragements and team work we have shared during this project.

Abstract

Toll fraud occurs whenever a perpetrator uses deception or dishonest means with the intention to receive telephony services free of charge or at a reduced rate. The introduction and increasing popularity of the internet has brought new possibilities like Voice over Internet Protocol (VoIP). Majority of the voice communication systems in most organizations nowadays have Internet Protocol (IP) connectivity to enable them leverage on the benefits of VoIP. A growing number of these organizations are however being targeted by toll fraud leading to losses estimated to run in billions of United States (US) dollars annually. Toll fraud therefore poses one of the largest threats to enterprise voice systems today.

This study presents a VoIP fraud detection model that identifies Artificial Neural Networks as an enabling tool to classify VoIP calls as either fraudulent or legitimate based on the attributes of the call. A multi-layer feed-forward neural network with back propagation learning algorithm was used to build the model. Actual Call Detail Records (CDR) collected from an operational VoIP system in Kenya was used. A total of 15,000 call records were sampled for the study. The data set contained time series of call records from both fraudulent and legitimate callers. Feature selection was performed on the data in order to eliminate redundant variables and select the attributes that would best describe fraudulent behavior. The sampled data was then labeled as being fraudulent or genuine based on the attributes of the call. The data set was partitioned as follows: 70% of the data set was set aside for training, 15% for validation and the remaining 15% was the testing set. The destination phone numbers fell into the following categories: on-net (Internal) and off-net (Mobile, National and International).The implementation of the Artificial Neural Network was based on the Matlab Neural Network toolbox. The trained network achieved 100% classification performance on the test data set. A web application for reading and classifying the call records in new CDR files was developed as part of this study. It utilized the trained neural network and provided a visualization of the classified records. The study established that Artificial Neural Network are a successful technology that can be applied in VoIP fraud detection since it was able to detect abrupt changes in established calling patterns which may be as a consequence of fraud. The implementation of the fraud detection tool will be a big step towards detection and mitigation of VoIP fraud.

Keywords: VoIP, Call Detail Record (CDR), Internet Protocol (IP), Artificial Neural Network,Matlab.

Table of contents

List of Figures:

List of Tables:

LIST OF ABBREVIATIONS AND ACRONYMS

WORD	**PHRASE**
VOIP	Voice over Internet Protocol
CFCA	Communications Fraud Control Association
IP	Internet Protocol
CDR	Call Detail Record
ANN	Artificial Neural Network
TESPOK	Telecommunication Service Providers of Kenya
PSTN	Public Switched Telephone Network
WAN	Wide Area Network
PBX	Private Branch Exchange
CPNI	Customer Proprietary Network Information
NOC	Network Operations Center
KDD	Knowledge Discovery in Databases
SEMMA	Sample Explore Modify Model and Assess

CHAPTER 1: INTRODUCTION

1.1 Background

Voice over Internet Protocol (VoIP) is a technology for communicating using Internet Protocol (IP) instead of traditional circuit-switched analog systems. VoIP applications have become even more widely used over the past two decades due rapid pace of development of IP-based technologies hence the migration from circuit-switched networks to packet-switched networks. VoIP reduces communication costs and provides enterprises with integrated services of voice and video applications. Due to these benefits, VoIP has seen rapid uptake in both the enterprise and consumer markets. VoIP fraud primarily occurs to a company with a weak network defense system. Billing systems and network vulnerabilities are easily exploited to gain access to a company's VoIP systems.

VoIP fraud (also toll fraud) in communications networks refers to any transmission of across a tele-communications network where the intent of the sender is to avoid or reduce legitimate call charges or force another party to pay(Johnson, 1996). VoIP Fraud affects any organization which uses or sells VoIP services. In most cases though, the fraud target is a business user. According to the Communications Fraud Control Association (CFCA) *Telecom Fraud Survey*, annual global telecom fraud losses amount to an estimated $54.4–$60 billion.

Traditionally, Africa has been a “Hot Continent” from telecom fraud, because the termination costs are very high and regulation is not as stringent as in other parts of the world (Transnexus, 2015). A 2011 study from the Communications Fraud Control Association (CFCA) found that the top five countries from which fraud originates are the United States, India, the United Kingdom, Pakistan, and the Philippines. The top five fraud terminating countries were Cuba, Somalia, Sierra Leone, Zimbabwe, and Latvia.

The *Kenya Cyber Security Report 2014* noted an increase in the number of VoIP attacks for Kenyan organizations with many affected organizations realizing that their VoIP servers had been hacked after huge bills from the VoIP service providers. The development of intelligent data analysis methods for fraud detection can therefore be motivated from an economic point of view. Toll fraud is an easy and profitable crime, since it is often undetected until large amounts of money have been lost. Often, the criminals start this activity at the beginning of a weekend, where the abuse is less likely to be noticed. The attack continues as long as the enterprise does not notice the abuse.

Unfortunately, most service providers will not notify an enterprise of potential toll fraud. If the enterprise only reviews reports at the end of the month or some other infrequent interval, the attack can go on for weeks, resulting in a significant expense to the enterprise. And since the calls were actually made from the enterprise, the service provider is typically unwilling to waive the charges, since providers also incur costs routing long distance and international calls.

Every time a call is completed on a telecommunications network, descriptive information about the call is saved as a *Call Detail Record (CDR).* CDR includes sufficient information to describe the important characteristics of each call. At a minimum, each CDR will include the originating and terminating phone numbers, the date and time of the call and the duration of the call. Telecommunications systems generate and store tremendous amount of CDR. CDRs constitute an enormous database within which useful knowledge about the caller can be extracted .The intentions of the callers are reflected in the observed CDR, which is used in describing the behavioral patterns of users. The call data can be used to learn models of calling behavior so that these models make inferences about callers' intentions since fraudulent call activity is discovered from anomalies in calling data and patterns. (Ogwueleka, 2009).

Traditional methods of data analysis have long been used to detect fraud; they require complex and time-consuming investigations that deal with different knowledge domains of like financial, economics, business practices and law (Nikita, 2015). The first industries to use data analysis techniques to prevent fraud were the telephony companies, the insurance companies and the banks (Decker, 1998). Fraud that involves cell phones, insurance claims, claims tax return, credit card transactions etc. represent significant problems for governments and businesses, but yet detecting and preventing fraud is still a challenge. Fraud is an adaptive crime, so it needs special methods of intelligent data analysis to detect and prevent it.These methods exists in the areas of data mining, machine learning and statistics etc. They offer applicable and successful solutions in different areas of fraud crimes.

1.2 Problem Statement

VoIP fraud occurs whenever a perpetrator uses deception to receive telephony services free of charge or at a reduced rate (Blavette,2001).Fraud detection refers to the attempt to detect illegitimate usage of a communications network(Michiakki,2010) . VoIP fraud has become one of the largest threats to enterprises VoIP systems today. While accurate cost estimates for toll fraud are difficult to pin down because many companies are reluctant to publicly admit they have been targeted, experts worldwide estimate the costs to run in the billions of dollars annually (Barson, 1996). According to the *Telecom Fraud Survey* from the Communications Fraud Control Association (CFCA), annual global telecom fraud losses amount to an estimated $54.4–$60 billion (USD). The 2014 report of the Telecommunication Service Providers of Kenya (TESPOK) reveals that many Kenyan companies have been victims of VoIP fraud and attributes large financial losses for firms to VoIP fraud.

In addition to financial losses, fraud may cause distress, loss of service, and loss of customer confidence (Hoath, 1998). As the popularity of VoIP continues to grow, the problem of VoIP fraud will become an increasing threat to the industry and the schemes of hacking the VoIP systems will continue to become more complex and powerful (The State of Phone Fraud, 1H 2012). With the continued migration from circuit-switched networks to packet-switched networks, it is expected that VoIP fraud will be worse, due to the openness and vulnerabilities associated with IP-based infrastructure. VoIP fraud is therefore a significant and growing problem in the telecommunications industry which needs to be detected and mitigated. Development of an effective VoIP fraud detection system is essential to minimize losses.

1.3 General Objective

The general objective of this research was to develop a Voice over IP fraud detection system using Artificial Neural Networks.

1.4 Specific Objectives

In order to achieve the aims of this research, we established the following key objectives:

1. To develop a neural network model for classification of VoIP call records as either fraudulent or genuine based on the attributes of the call.
2. To develop a web application for classifying VoIP call records using the neural network model.

1.5 Justification

VoIP technologies have become more accessible and popular with the increasing adoption of IP networks, fraud attacks have grown as well. Fraudulent VoIP activity across IP networks is increasing, and will continue to be a big challenge for service providers and enterprises in the coming years (Kenya cyber Security Report, 2014). Large financial losses for businesses in the region are being attributed to VoIP fraud. In addition to financial losses, fraud may cause distress, loss of service, and loss of customer confidence (Hoath, 1998).VoIP fraud is therefore a significant and growing problem in the telecommunications industry which needs to be detected and mitigated.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

VoIP is a technology that enables the delivery of voice communications and multimedia sessions over IP networks at low cost or no cost at all. With continuous progress being made in network technology, VoIP applications have become ever more widely used. VoIP reduces communication costs and provides enterprises with integrated services of voice and video applications. VoIP is a key enabling technology for the migration of circuit-switched PSTN architectures to packet based networks.

Figure 2-1 below highlights the components of a VoIP system-VoIP allows users to make a call directly from a computer, IP phone, or a traditional phone connected to a special adapter. In addition, wireless "hot spots" in locations such as airports, parks, and cafes allow you to connect to the Internet and may enable you to use VoIP service wirelessly. VoIP converts voice into a digital signal that travels over an IP network (or the Internet).

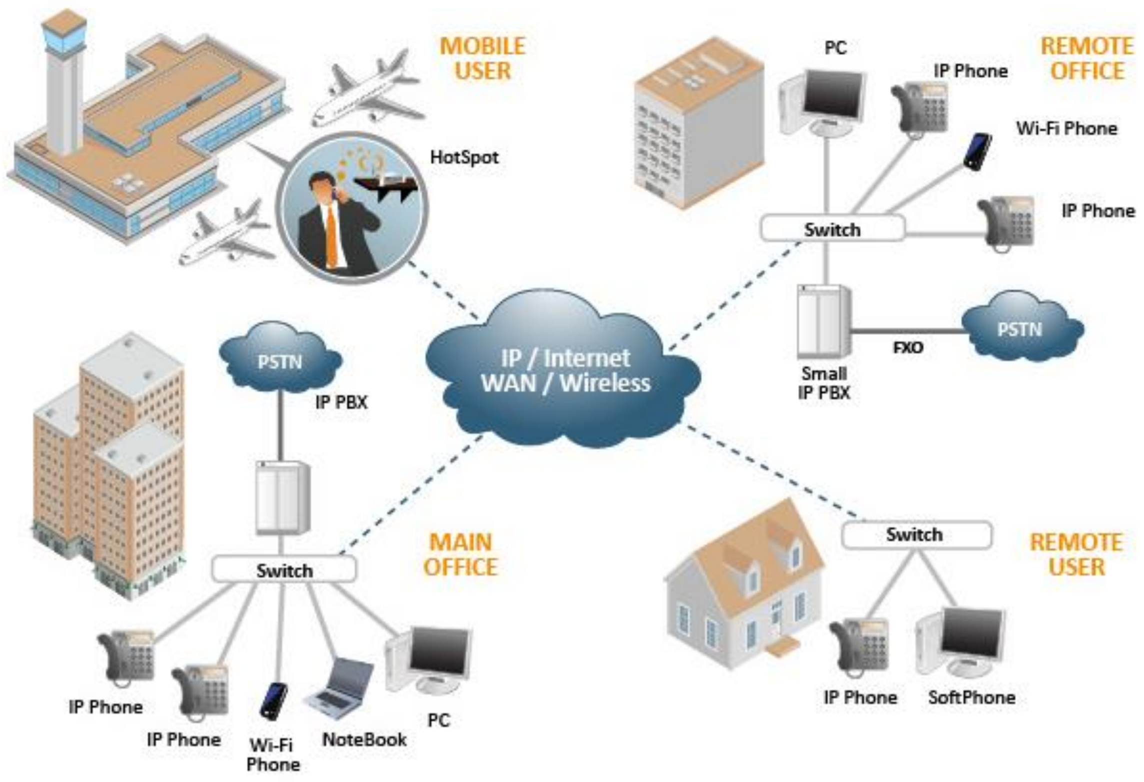


Figure 2-1: Voice over IP system schematic

VoIP is a technology that enables the delivery of voice communications and multimedia sessions over IP networks. VoIP permits the integration of data, voice, and video into one communication channel. The term digital convergence refers to this phenomenon of multiple media delivered over a single network. Some of the applications and services include video conferencing, live webcasting, video streaming, collaboration and team management software, security surveillance, contact center applications, remote multimedia solutions and unified messaging (Bidoli,2006).

2.2 Benefits of VoIP

i. **Cost savings**: When we move away from PSTN, long-distance phone calls become inexpensive. Instead of being processed across conventional commercial telecommunications line configurations, voice traffic travels on the Internet or over private data network lines. For the enterprise, VoIP reduces cost for equipment, lines, manpower, and maintenance. For consumers, VoIP reduces the charge of subscription or usage, especially for long distance and international calls.

ii. **No geographical boundary:** The VoIP service area becomes virtualized without geographical limit. That is, the area code or country code is no longer bound to a specific location.

iii. **Rich media service:** The legacy phone system mainly provides voice and fax service. VoIP technology makes rich media services possible by integrating with other protocols and applications-instant messages make voice or video calls e.t.c.

iv. **Service mobility**: The context of mobility here includes service mobility as well. Wherever the phone goes, the same services could be available, such as call features, voicemail access, call logs, security features, service policy, and so on.

v. **Toll by-pass**: Call routing and path selection to route inter office calls over WAN links with PSTN fallback

2.3 VoIP Fraud

VoIP fraud occurs whenever a perpetrator uses deception to receive telephony services free of charge or at a reduced rate (Blavette, 2001). VoIP fraud generally involves a third party making long duration international calls at the expense of a business. Hackers gain unauthorized access the business’s phone system and generate profit from the calls that they make to international premium rate numbers, leaving the business with losses running into several Millions of shillings when such attacks go undetected and remediated. VoIP fraud is a significant and growing problem in the telecommunications industry.

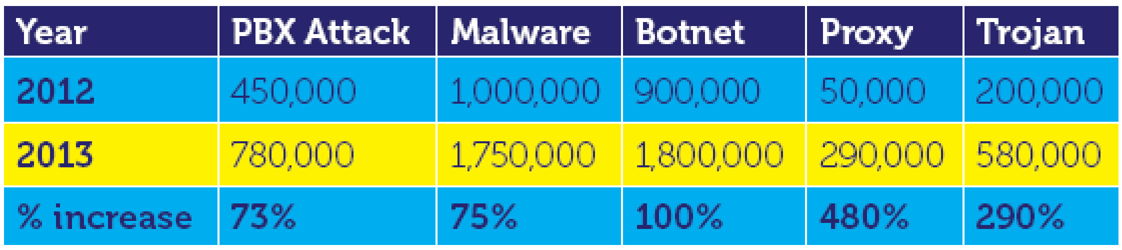
Because fraudsters often attack during weekends, fraud events often go undetected for many hours. A single fraud event can easily cost a company between three and fifty thousand dollars. In many cases, this number can be even larger. A 2009 attack on an Australian company's VoIP PBX resulted in 11,000 international calls in just 46 hours, leaving the SIP provider with a bill in excess of $120,000. A 2011 weekend episode in South Africa resulted in a bill of over $12,000 and another in the US cost victims more than $1.4 million.

The introduction and increasing popularity of VoIP services has led to the majority of PBX's used in organizations to have IP connectivity. With this migration from circuit-switched networks to packet-switched networks, it is expected that VoIP fraud will be worse, due to the openness and vulnerabilities associated with IP-based infrastructure This has increased the number of attack vectors available to be exploited where they have not been secured. A growing number of businesses are being targeted by VoIP fraud.

A number of recent publicized toll fraud attacks highlight this problem. For example:

- A hacker broke into the FEMA voicemail system and caused about $12,000 in charges for calls to the Middle East and Asia.
- Hackers stole over $120,000 in fraudulent international call charges from a small business in Perth, Australia.
- A hacked business in the Greater Toronto area was subjected to a bill for over $207, 000 of fraudulent calling charges for calls to Sierra Leone.
- Hackers used a library phone system in Duxbury, MA, to steal $15,000 worth of international charges for calls to India, the Philippines, and Jordan.

From Figure 2-2 below shows the threat landscape in Kenya, for the period under review the number of PBX (VoIP) attacks increased by 73% from 450,000 attacks detected in 2012 to 780,000 attacks detected in 2013.

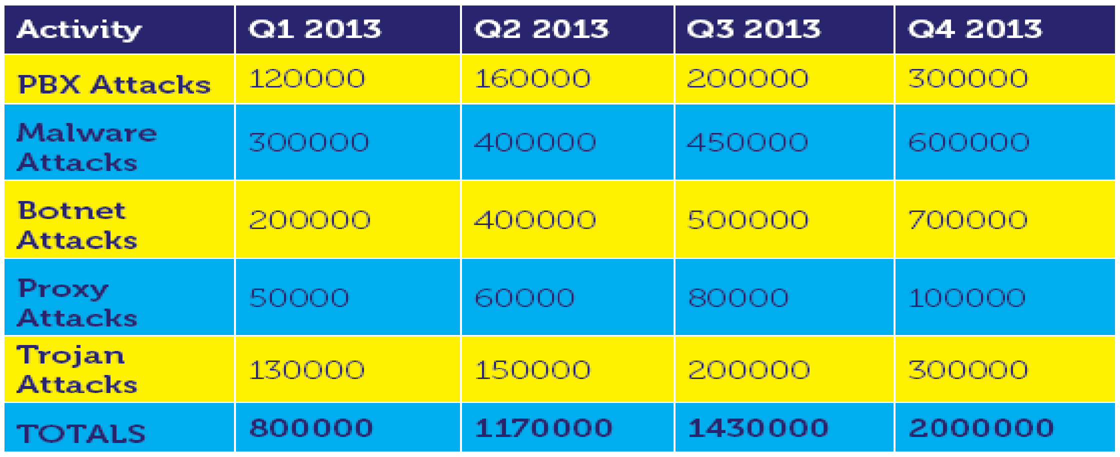


Year	PBX Attack	Malware	Botnet	Proxy	Trojan
2012	450,000	1,000,000	900,000	50,000	200,000
2013	780,000	1,750,000	1,800,000	290,000	580,000
% increase	73%	75%	100%	480%	290%

Figure 2-2: Threat Landscape 2012 vs. 2013

(Source: Kenya Cyber Security Report 2014)

Figure 2-3 below indicates an increasing number of VoIP attacks in Kenya as compared to the attacks for the period under review.



Activity	Q1 2013	Q2 2013	Q3 2013	Q4 2013
PBX Attacks	120000	160000	200000	300000
Malware Attacks	300000	400000	450000	600000
Botnet Attacks	200000	400000	500000	700000
Proxy Attacks	50000	60000	80000	100000
Trojan Attacks	130000	150000	200000	300000
TOTALS	800000	1170000	1430000	2000000

Figure 2-3: Threat Analysis

(Source: Kenya Cyber Security Report 2014).

2.4 Data Mining In Relation to VoIP

Data mining is the process of analyzing and automatically extracting interesting and previously unknown dependencies and relationships among data, leading to a better understanding. Data mining uses defined methods for analyzing current and historical data in order to predict future trends. Data mining techniques can be applied for finding hidden correlations, patterns and unexpected trends in very large datasets. Data mining process is used in a large variety of activities in business, science and engineering to make better business decisions and strategies, by discovering patterns and relationships in the database.

Over a period of time, an individual's telephone generates a large pattern of use. The pattern of use could include international calls and time-varying call patterns among others. Anomalous use can be detected within the overall pattern; like subscribers abuse of free call services such as emergency services (Ogwueleka,2009). A descriptive analysis of the call data for each telephone user can be used for knowledge extraction. Interpretation by way of clustering or grouping similar patterns can help in isolating suspicious call behavior within the telecommunication network. This can also help fraud analysts in their further investigation and call pattern analysis of subscribers.

The problem of revealing users calling network profile and statistical calling information based on their calls is essential in various applications. Most of the research in this domain shows that in the telecommunication industry the data mining techniques can be used with success in application like: market management, fraud detection and customer profiling.

Below is a list of features that one might use when generating a summary description of a customer based on the calls they originate and receive over some time period P:

- Average call duration
- Number of answered calls
- Calls to/from a different area code
- Number of weekday calls (Monday – Friday)
- Number of daytime calls (9am – 5pm)
- Average number of calls received per day
- Average number of calls originated per day
- Number of unique area codes called during P

These eight features can be used to build a customer profile. Such a profile has many potential applications. For example, it could be used to distinguish between business and residential customers based on the percentage of weekday and daytime calls. Most of the eight features listed above were generated in a straight forward manner from the underlying data, but some features, such as the eighth feature, required a little more thought and creativity. The above example demonstrates that generating useful features, including summary features, is a critical step within the data mining process. Should poor features be generated, data mining will not be successful.

Although the construction of these features may be guided by common sense and expert knowledge, it should include exploratory data analysis. For example, the use of the time period 8am-5pm in the fifth feature is based on the commonsense knowledge that the typical workday is 8 to 5 (and hence this feature may be useful in distinguishing between business and residential calling patterns).

For some applications, such as fraud detection, the summary descriptions, sometimes called signatures must be updated in real-time for millions of phone lines (Cortes, 2001). This requires the use of fairly short and simple summary features that can be updated quickly and efficiently.

2.5 Artificial Neural Networks

An Artificial Neutral Network (ANN) is a system that is based on the biological neural network, such as the brain. The Neural Network is ideally composed of three layers, the input layer, the hidden layer, and the output layer show in Figure 2-4 below. The input layer consists of input nodes which represent the system's variable. The hidden layer consists of nodes which facilitate the flow of information from the input to the output layers. The flow is controlled by weight factors associated with each connector. The output layer consists of nodes which represent the system's classification decision. The values of the output nodes are compared with cutoffs to determine the output and classify each case. The weight adjustment is known as training. The training process consists of running input values over the network with predefined classification output nodes. This process runs until the weight values are minimized to an error function. Testing samples are used to verify the performance of the trained network. This type of computational algorithm can be applicable in linear classification, where an on or off (yes or no) is the ultimate result, depending on evaluation of the inputs. ANNs can also be used in regression problems where there is need to obtain a real-valued target. According to Russell et al. (2009), neural networks remain the most popular and effective learning systems in the field of Artificial Intelligence (AI). An ANN is comprised of a network of artificial neurons (also known as "nodes").

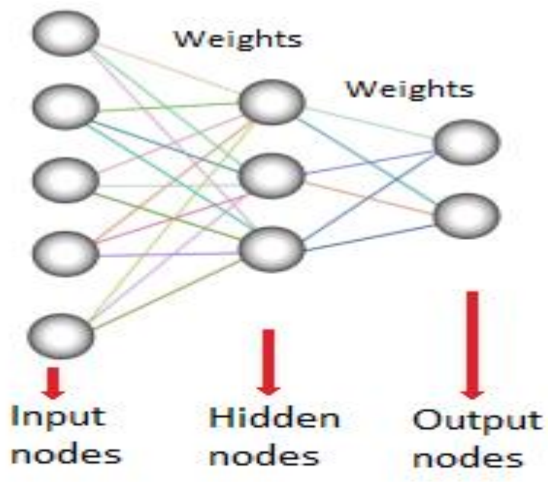


Figure 2-4: Representation of an Artificial Neural Network.

ANNs are popular is due to their robustness, fault tolerance, ability to learn and generalize adaptability, universal function approximation and parallel data processing. This enables them to solve complex non-linear and multi input-output (IO) relationship problems

Ortiz-Rodriguez et al., (2013) and Gomes et al. (2011) also states that ANNs are useful in practical applications due to their ability to do non-linear mapping, parallel processing methodology, ability to learn from the environment and their subsequent adaptability to the environment. Ghaffari et al. (2006) state that when compared to other methods, ANN have been shown to be superior as modeling technique for data sets with non-linear relationships. This enables them to be applied for data fitting and prediction. Russell et al. (2009) says that modern neural network research has laid focus in two areas i.e. making effective network architectures, algorithms and understanding of their mathematical properties and secondly, modeling empirical properties of actual neurons and collections. With time, developments in the AI field of neural networks have enabled these systems to be comparable with corresponding techniques from statistics, pattern recognition and machine learning. AI is now able to solve the problem of expressing knowledge needed by a system by applying learning methods instead of hard coded knowledge. In another study, Cerna et al. (2005) stated that the most popular types of ANN were Kohonen network (Self Organizing Map) and Multi-Layer perceptron (MLP). The former employs unsupervised learning to cluster data while MLP uses supervised learning to create a classifier. The classifier is then able to match outputs (responses) to inputs (predictors) in the training data. The trained classifier can then predict outputs for new cases. These studies above suggest that ANNs are useful in two AI fields – classification and regression. In classification, the interest is in placing members into distinct groups, while regression involves formulating a real-value target function for any particular input function.

It is possible to apply the various algorithms in implementing ANNs i.e. unsupervised, supervised or reinforcement. It is also possible to have a network that is feed-forward or recurrent. The choice of parameters is application dependent. Training or learning is just the process of 'determining an optimized set of weights based on the statistics of the examples' (Smith, 1997). The purpose of training is therefore to establish a set of network weights, for each node, that can properly map any input data into the desired output. Learning enables us to estimate a set of parameters, say w, to define the hypothesis hw(x). After attaining the hypothesis, the training set can be abandoned, having all been summarized by w. This type of learning is called parametric model learning. Learning generally formulates a methodology of adjusting the weight, w, to fit solving of problems for the neural network. Additionally, training can be done using either of the following methods - Direct training, by adjusting weights or Indirect training, by adjusting the threshold values.

For practical ANNs, there is usually the need to adjust the weights of all neurons automatically, after obtaining the final output. There are different methods of weight adjustment (training). These include simulated annealing, resilient propagation and back propagation amongst others. Ghaffari et al. (2006) compares various methods used in training ANNs and goes ahead to make comparison about the three broad classes that they identified for their application. These training methods are: Gradient descent back propagation algorithm (using incremental or batch back propagation), Quasi-Newton/Levenberg-Marquardt (LM) back propagation algorithm and lastly, Genetic algorithm (GA).Their findings ranked the order of predictive abilities, from best, which was similar to the list above. Apart from their research showing that back propagation was the best performing, they also state that this method is the most commonly used in many applications. Batch Gradient Descent computes weight updates after summing them over all the training examples (also called 'offline' learning). Incremental Gradient Descent on the other hand applies 'online' learning, where the weights are updated after each training example. There are many other variants of ANN training methods developed by different people. Gomes et al. (2011) singles out LM and Fletcher-Reeves conjugate algorithm in their study.

Russell et al. (2009) recommends that one needs to just try several settings and keep the best. That involves training on a particular setting and testing the efficacy of the settings, then repeating for different settings before choosing the most optimum. One approach, the optimal brain damage, starts by training a fully connected network, then dropping units while observing the performance after retraining. A converse approach is the tiling algorithm, which trains from a single unit and is progressively built up with additional units while monitoring its performance.

There are several issues to consider in the design of ANNs. These issues are: Type of network, Type of training, Size of Training, Validation and Testing data sets, Number of input and output units, Number and size of hidden layers, Number of repetitions during training (epoch), Choice of activation function and Size of data set:

2.5.1 Type of Network

ANNs can be feedforward or recurrent. There is also need to determine the method of adjusting the neuron weights, which is usually by backpropagation. Backpropagation calculates the value of errors from output towards input neurons, with a view of minimizing the error. Neurons contributing to higher degree of error are also subjected to higher degree of error correction. A variant of backpropagation, called resilient propagation, tries to avoid the possibility of the ANN getting stuck on a local minima by using only the sign of the error gradient as positive, zero or negative. Applications that can use backpropagation are also candidates for resilient propagation algorithms (Code Project, 2011).

2.5.2 Training an Artificial Network

There are two approaches to training: supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Unsupervised training is where the network has to make sense of the inputs without outside help. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data.

2.5.3 Size of Training, Validation and Testing Sets

The ANN learning process should be exposed to as much of the available data as possible, instead of relying on a small fraction, Russell et al. (2009). In their paper, Ortiz-Rodriguez et al. (2013) state that their methodology of Robust Design of Artificial Neural Networks (RDANN) has enabled them to propose training set of 80% of the data and the balance 20% for testing. In considering data sets and their efficacy, Keogh et al. (2003) suggest that a wide dataset is necessary in testing any algorithms, with one subset for fine tuning an approach, while the other set for actual tests. They also advocate for designs that are free of implementation biases, so that such can be applied to different datasets to realize similar results. These best practices should be applicable to any time series data mining research in areas such as indexing, clustering, classification or segmentation. The need for large datasets is also noted by Giles et al. (2001), who state that many models could fit a training set very well, though only few would ultimately provide a good enough generalized solution. Ben-David et al. (2010) looked at the issue of training and testing sets and the effect of data volumes in these sets. Their work also tried to answer the question on what would happen if the target domain was little known (such as prediction based on testing set)or unknown (such as language processing). They concluded that several mathematical relationships can assist in achieving an optimum level of training sets that guarantees low error rates. Therefore, ultimately, a choice needs to be made on what to use based on the problem and how efficient the solution can be, however, a training set of 70% to 80% is considered optimal.

2.5.4 Input, Hidden and Output units

The input layer is composed of the values in a data record, which constitute inputs to the next layer of neurons. The next layer is called a hidden layer; it defines the number of hidden layers to be implemented on the network before the training process is started. The number of the hidden layers is determined by the nature of data used in the training process. The final layer is the output layer, where there is one node for each class.

In proposing designs for evolutionary neural networks, Han et al.(2006) contends that application of NNs to real world problems needs a delicate balance on the topology of the ANN, which is usually problem specific. They argue that repeating 'trial and error' cycles based on previous experiences of similar problems may be the way out. However, their Evolutionary Neural Network (ENN) design tries to automatically conFigure the topology of the NN. Their most optimal design had 10 inputs, 15 hidden nodes and 2 output nodes. On their part, Ortiz-Rodriguez et al. (2013) used their RDANN method to suggest a configuration of 7:14:31, instead of 'trial and error'. Other considerations for deciding the number of hidden neurons include: A number between number of input neurons and number of output neurons, Sum of output neurons and two-thirds the size of input neurons or less than two-times the input neurons (Heaton Research, 2013c).

2.5.5 Number of Training Cycles

This is number of cycles the training data is passed through the network. The more the training cycles the more the accuracy. Training involves feeding the input pattern to the network in order to generate an output pattern, then comparing the generate output with the actual true output to determine an error value. The next data set is fed to the network and neuron weights adjusted with a view of reducing the error. This process is repeated to the end of the dataset, then back to the start of dataset, until the error is significantly small. A run through a complete dataset is called an epoch.

2.5.6 Structuring the Network

The number of layers and the number of processing elements per layer are important decisions. These parameters to a feedforward, back-propagation topology are also the most ethereal - they are the "art" of the network designer. There is no quantifiable, best answer to the layout of the network for any particular application. There are only general rules picked up over time and followed by most researchers and engineers applying this architecture to their problems (Code Project, 2011).

Rule One: As the complexity in the relationship between the input data and the desired output increases, the number of the processing elements in the hidden layer should also increase.

Rule Two: If the process being modeled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization of the training set, and not a true general solution effective with other data.

Rule Three: The amount of training data available sets an upper bound for the number of processing elements in the hidden layer(s). To calculate this upper bound, use the number of cases in the training data set and divide that number by the sum of the number of nodes in the input and output layers in the network. Then divide that result again by a scaling factor between five and ten. Larger scaling factors are used for relatively less noisy data. If you use too many artificial neurons the training set will be memorized. If that happens, generalization of the data will not occur, making the network useless on new data sets.

2.6 Overview of Related Works

Machine learning provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data. The process of machine learning is similar to that of data mining since they both search through data to look for patterns. In data mining, the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data. Supervised learning is the task of inferring a function from labeled training data. The training data consist of a set of training examples. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. Different models can be used to detect fraudulent behavior in VoIP environments (Graaff, 2011).

2.6.1 Supervised Learning

Supervised learning models are trained with Call Detail record that have been pre-classified as either fraudulent or non-fraudulent (Frank et all, 1999).After training future calls made will then be classified as fraudulent or not. Care needs to be taken in order to ensure that the training data is correctly classified to prevent classifying future legitimate incorrectly as illegitimate (Moreau et al, 1996).The model can be seen as a rule to classify calls.

A disadvantage of this model is the training data needs classifications of both types and the model can only detect a fraudulent call if it has previously occurred in the training data. Since legitimate calls occur more often than fraudulent calls, the training data will mostly contain legitimate calls, leading to a misclassification of the model (Bolton et al, 2001).

For the model to adapt to future fraudulent calls, it can be retrained with new training data. This new training data consists of the old training data and the new training data which h could contain new types of fraudulent call. This approach is impractical since the training data size will become too large and training will take too much time.

Barson (1996) use feed-forward neural networks based on supervised learning to detect mobile phone fraud in their simulated database of call records. They simulate six types of users ranging from low use local users to high use international business users. They report their neural network classifier to correctly classify 92.5% of the calling data. Their work does not include any comment on the false alarm probability and is also based on simulated data.

Moreau et al. (1997) report fraud detection in a real mobile communication networks. Their approach is based on feed-forward neural networks with supervised learning. They use different user-profiles and also consider comparisons between past and present behavior. They conclude that although their work is in a prototype phase.

A different approach to detect future fraudulent calls will be to train another model, using the same machine learning algorithm and adding it as a rule to previous training models to form a rule set. Incoming calls are then evaluated by each rule in the rule set, if one rule classifies the call as fraudulent, it needs not be evaluated by other rules.

If no rule classified the call as fraudulent, it most probably is not fraudulent, or the specific rule to detect the specific fraudulent call has not been modeled yet.

2.6.2 Unsupervised Learning

Most of the time, training data has no pre-classification of being fraudulent or not, in such cases, unsupervised models are used to find groups in the training data, summarizing the behavior of the system to detect any future calls from the groups formed(Bolton et al,2001).. An advantage of using unsupervised models over supervised models is that undiscovered types of fraud may be detected, where supervised models can only detect fraudulent patterns they were trained on.

Unsupervised models can be used to cluster data based on statistical information of each account to form groups of accounts with similar characteristics. If an account has always been part of a specific group, it will be excluded from its original group if it contains a fraudulent call, forming part of another group or no group (Bolton et al, 2001).

This fraudulent account can be seen as a local outlier to its group. Unsupervised models can be used to track account behavior. For account behavior modeling certain time period of calls (sliding window) in an account is used for training to model the behavior of the account(Bolton et al,2001). A future call is then compared with the modeled behavior window to indicate any change in the calling behavior. The window is then slid, the latest call made forming part of the sliding window and the oldest call is removed from the window.

Rosset et al, (1999) proposed a rule-discovery framework for fraud detection, in which candidate rules are identified first and the most relevant rules are selected on the basis of a suggested algorithm. (Ruiz-Agundez et al. 2010) proposed a rule-based fraud detection framework for VoIP services. In the proposed framework, a rule engine is generated using a knowledge base. (Olszewski, 2012) attempted to construct user profiles on the basis of Kullback-Leibler divergence to prevent fraud detection.

Rule-based approaches have been previously employed in toll fraud prevention. A rule-based approach uses predefined rules developed by experts. A notification is triggered when a rule is satisfied. As long as an effective set of rules can be defined, the rule-based method can be effective against fraud attacks. This method is ineffective for unknown types of fraud (S. Kim et al, 2013).

2.7 Privacy Concerns

Data mining applications must always consider privacy issues. This is especially true in the telecommunications industry, since telecommunication companies maintain highly private information, such as whom each customer calls. Most telecommunication companies utilize this information conscientiously and consequently privacy concerns have thus far been minimized. A more significant issue in the telecommunications industry relates to specific legal restrictions on how data may be used. In the United States for example, the information that a telecommunications company acquires about their subscribers is referred to as Customer Proprietary Network Information (CPNI) and there are specific restrictions on how this data may be used. The Telecommunications Act of 1996, along with more recent clarifications from the Federal Communications Commission, generally prohibits the use of that information without customer permission, even for the purpose of marketing the customers other services. In Kenya, the Kenya Information and Communications Act of 2009 articulates the right to privacy while using telecommunications equipment and/or services.

In the case of customers who switch to other service providers, the original service provider is prohibited from using the information to try to get the customer back (e.g., by only targeting profitable customers). Furthermore, companies are prohibited from using data from one type of service (e.g., wireless) in order to sell another service (e.g. landline services). Thus, the use of data mining is restricted in that there are many instances in which useful knowledge extracted by the data mining process cannot be legally exploited. Much of the rationale for these prohibitions relates to competition. For example, if a large company can leverage the data associated with one service to sell another service, then companies that provide fewer services would be at a competitive disadvantage.

CHAPTER 3: METHODOLOGY

3.1 Introduction

Several data mining process models have been developed over the years among them Knowledge Discovery in Databases (KDD), Sample Explore Modify Model and Assess (SEMMA), and Cross Industry Process Model for Data Mining (CRISP-DM) (Azevedo, 2008). These process models provide steps that guide a data mining solution.

This project employed the CRISP-DM process model in the implementation of the VoIP fraud detection system. The CRISP-DM model has been recommended as the best model by various data miners as it encourages best practices and offers organizations the structure needed to realize better, faster results from data mining (Shearer, 2000).The CRISP-DM methodology employs six steps as illustrated in Figure 3-1 below. Each of the steps is described in more detail in the section:

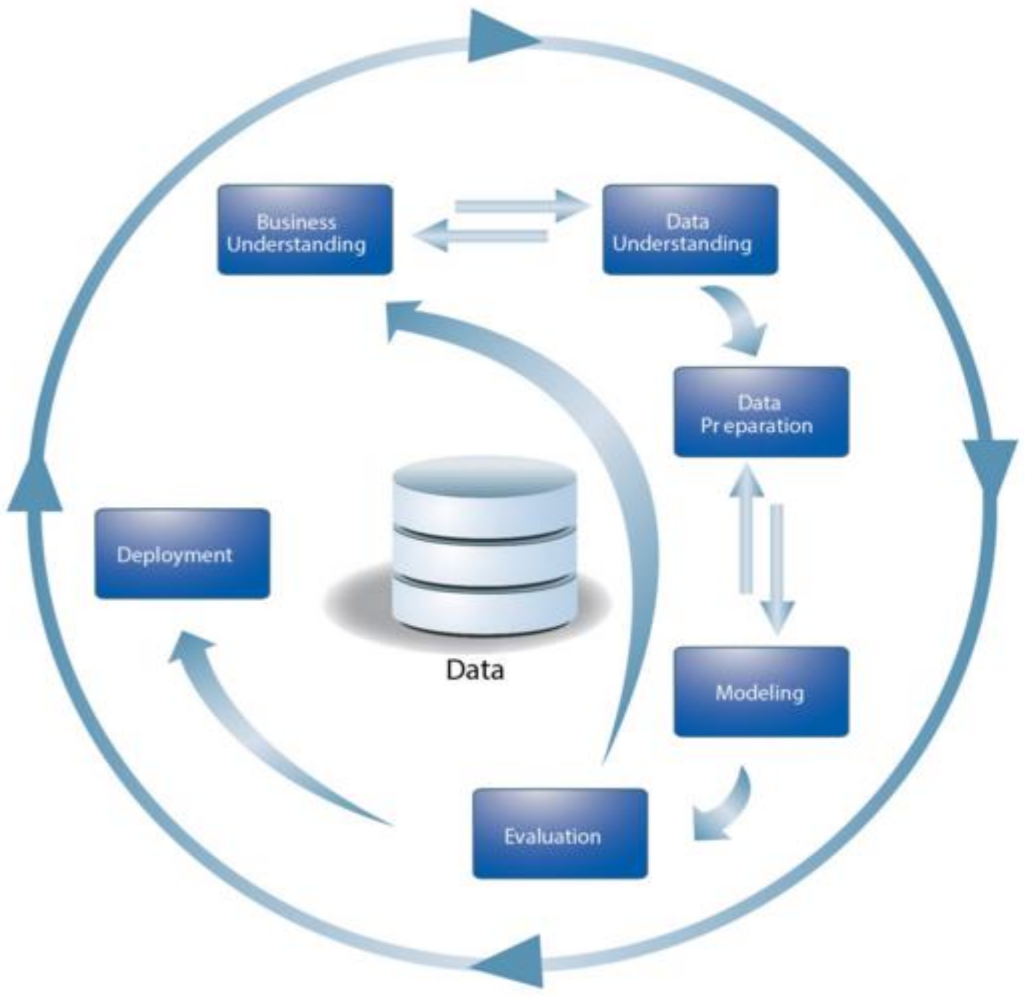


Figure 3-1: The CRISP-DM Process Model

The steps employed in the CRISP-DM model are described below:-

1. **Business understanding** – this phase entails understanding the objectives and the requirements of a project, translating the project into a data mining problem and translating of the project objectives into a plan.

2. **Data understanding** – this entails data collection, identification of data quality problems, and identification of subsets that hypotheses can be formed about.

3. **Data preparation** – involves extraction and transformation of data into a form that will be loaded into the modeling tool. Also entails selection of the most important attributes.

4. **Modeling** – this step entails selection of appropriate modeling techniques and their application. Different modeling techniques will normally have specific requirements on the data as a result of which a return to the data preparation stage may be necessary.

5. **Evaluation** – the model developed is thoroughly evaluated to ensure it captures all the issues of the problem to be solved and achieves all the objectives of the project.

6. **Deploymen**t – in this step the results of the data mining process are interpreted and presented using reports.

CRISP-DM as an approach to knowledge discovery in databases or data mining has been defined as a standard by the consortium of companies that formed it (Chapman et. al 2000).This standard contains detailed descriptions of all the tasks that are undertaken under each phase and the corresponding outputs from each phase task. These phases and tasks are briefly highlighted in the table 3-1 below:-

Table 3-1: Phases and tasks of the CRISP-DM process model

PHASE	**TASKS**	**OUTPUT**
Business Understanding	− Determine business objectives − Assess the situation − Determine data mining goals − Produce project plan	− Business objectives −Business success criteria − Inventory of resources − Requirements, assumptions and constraints − Data mining goals − Project plan

Data Understanding	− Collect initial data − Describe initial data − Explore data − Verify data quality	− Data collection report − Data description report − Data exploration report − Data quality report
Data Preparation	− Select data − Clean data − Construct data − Integrate data − Format data	− Rationale for inclusion/exclusion of data − Data cleaning report − Derived attributes − Generated records − Merged data − Reformatted data
Modeling	− Select modeling techniques − Generate test design − Build model − Assess model	− Modeling techniques − Test design − Parameter settings − Model assessment, revised parameter settings
Evaluation	− Evaluate results − Review process − Determine next steps	− Assessment of results with respect to business success criteria − Approved models − Review of process − List of possible actions
Deployment	− Plan deployment − Plan monitoring and maintenance − Produce final report − Review project	− Deployment plan − Monitoring and maintenance plan − Final report

3.2 Data Sourcing

Successful data mining projects usually employ the use of large collections of data normally subject oriented, historical and time variant. The goal for successful data mining is always to ensure the data covers all possible phases of change in the subject being investigated. In order to meet this requirement we aim to use data collected over a specific period of time containing all possible scenario.

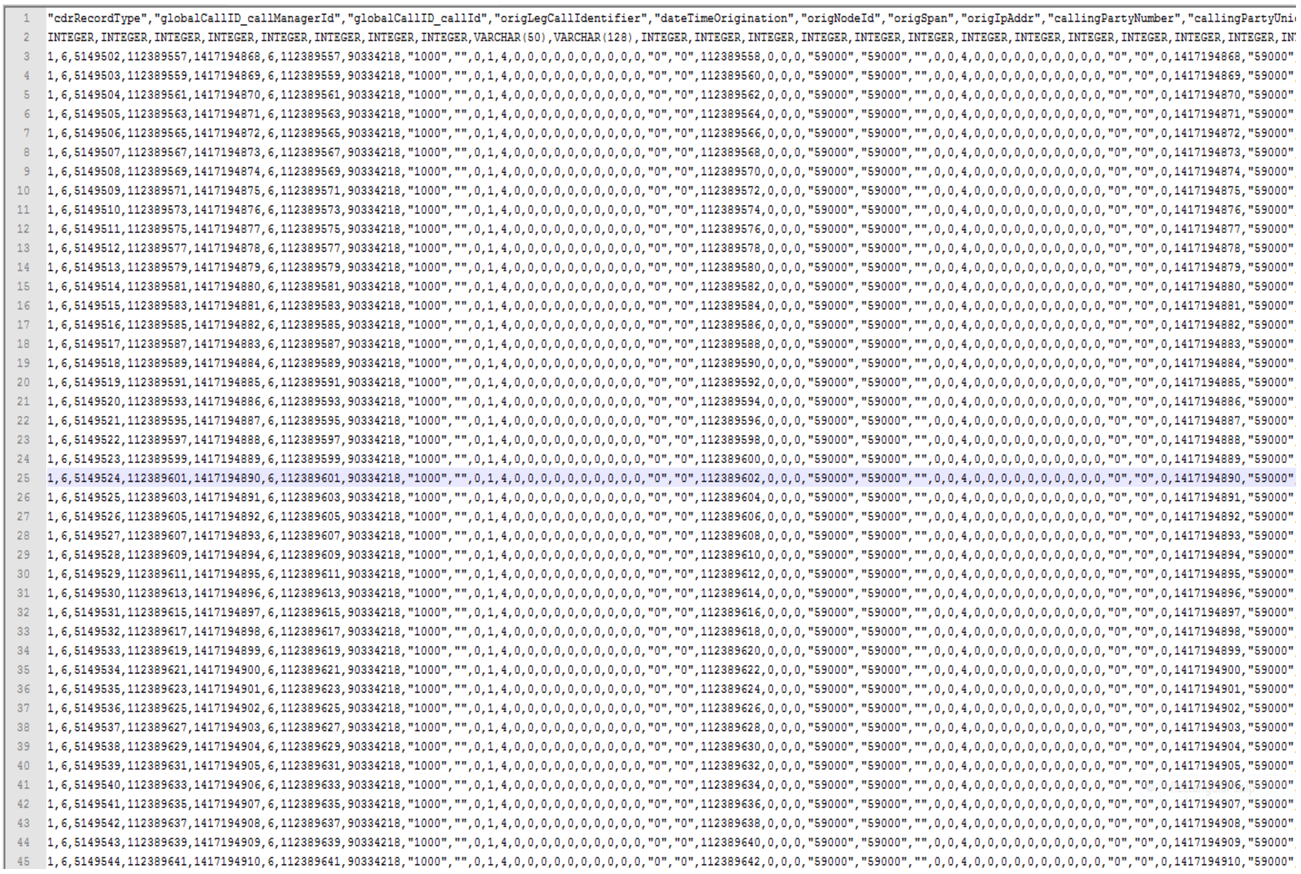
For this study, actual Call Detail Records (CDR) collected over three months from an organization in Kenya that uses VoIP was used. The organization's VoIP system generated CDR dump in every 15 minutes therefore a total of 96 dumps were available per day. The three months' data provided a total of 24,694 CDR files and 710,238 records. Each CDR file had a total of 104 fields including the originating and destination phone numbers, the date and time of the call and the duration of the call. CDR is a data record that contains information related to a telephone call, such as the origination and destination addresses of the call, the time the call started and ended, the duration of the call, the time of day the call was made and any toll charges that were added through the network or charges for operator services, among other details of the call.

At minimum, each CDR dump included the originating and destination phone numbers, the date and time of the call and the duration of the call. The destination phone numbers falls into the following categories: Internal (on-net), Mobile, National and international. The call data will additionally include the following call scenarios: Conference calls, transferred calls etc.

The data set contains time series of call records from both fraudulent and legitimate callers and will be used in developing the neural network model. For privacy reasons the results will be presented with masked call data.

The data set was partitioned into three parts: training set, validation set and a test set. The training set was used in training the network. The Validation set was used to fine-tune model. Finally, the test set was used in testing the accuracy of the model.

Figure 3-2 below shows a section of the raw CDR data. The first row is a description of the attribute while row 2 indicates the possible range of values e.g. integer, text string, variable character and so on. The actual data begins on the third row.



```
1  "cdrRecordType","globalCallID_callManagerId","globalCallID_callId","origLegCallIdentifier","dateTimeOrigination","origNodeId","origSpan","origIpAddr","callingPartyNumber","callingPartyUni
2  INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,VARCHAR(50),VARCHAR(128),INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,IN
3  1,6,5149502,112389557,1417194868,6,112389557,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389558,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194868,"59000"
4  1,6,5149503,112389559,1417194869,6,112389559,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389560,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194869,"59000"
5  1,6,5149504,112389561,1417194870,6,112389561,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389562,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194870,"59000"
6  1,6,5149505,112389563,1417194871,6,112389563,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389564,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194871,"59000"
7  1,6,5149506,112389565,1417194872,6,112389565,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389566,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194872,"59000"
8  1,6,5149507,112389567,1417194873,6,112389567,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389568,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194873,"59000"
9  1,6,5149508,112389569,1417194874,6,112389569,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389570,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194874,"59000"
10 1,6,5149509,112389571,1417194875,6,112389571,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389572,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194875,"59000"
11 1,6,5149510,112389573,1417194876,6,112389573,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389574,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194876,"59000"
12 1,6,5149511,112389575,1417194877,6,112389575,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389576,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194877,"59000"
13 1,6,5149512,112389577,1417194878,6,112389577,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389578,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194878,"59000"
14 1,6,5149513,112389579,1417194879,6,112389579,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389580,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194879,"59000"
15 1,6,5149514,112389581,1417194880,6,112389581,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389582,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194880,"59000"
16 1,6,5149515,112389583,1417194881,6,112389583,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389584,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194881,"59000"
17 1,6,5149516,112389585,1417194882,6,112389585,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389586,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194882,"59000"
18 1,6,5149517,112389587,1417194883,6,112389587,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389588,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194883,"59000"
19 1,6,5149518,112389589,1417194884,6,112389589,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389590,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194884,"59000"
20 1,6,5149519,112389591,1417194885,6,112389591,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389592,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194885,"59000"
21 1,6,5149520,112389593,1417194886,6,112389593,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389594,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194886,"59000"
22 1,6,5149521,112389595,1417194887,6,112389595,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389596,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194887,"59000"
23 1,6,5149522,112389597,1417194888,6,112389597,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389598,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194888,"59000"
24 1,6,5149523,112389599,1417194889,6,112389599,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389600,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194889,"59000"
25 1,6,5149524,112389601,1417194890,6,112389601,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389602,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194890,"59000"
26 1,6,5149525,112389603,1417194891,6,112389603,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389604,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194891,"59000"
27 1,6,5149526,112389605,1417194892,6,112389605,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389606,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194892,"59000"
28 1,6,5149527,112389607,1417194893,6,112389607,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389608,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194893,"59000"
29 1,6,5149528,112389609,1417194894,6,112389609,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389610,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194894,"59000"
30 1,6,5149529,112389611,1417194895,6,112389611,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389612,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194895,"59000"
31 1,6,5149530,112389613,1417194896,6,112389613,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389614,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194896,"59000"
32 1,6,5149531,112389615,1417194897,6,112389615,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389616,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194897,"59000"
33 1,6,5149532,112389617,1417194898,6,112389617,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389618,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194898,"59000"
34 1,6,5149533,112389619,1417194899,6,112389619,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389620,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194899,"59000"
35 1,6,5149534,112389621,1417194900,6,112389621,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389622,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194900,"59000"
36 1,6,5149535,112389623,1417194901,6,112389623,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389624,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194901,"59000"
37 1,6,5149536,112389625,1417194902,6,112389625,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389626,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194902,"59000"
38 1,6,5149537,112389627,1417194903,6,112389627,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389628,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194903,"59000"
39 1,6,5149538,112389629,1417194904,6,112389629,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389630,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194904,"59000"
40 1,6,5149539,112389631,1417194905,6,112389631,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389632,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194905,"59000"
41 1,6,5149540,112389633,1417194906,6,112389633,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389634,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194906,"59000"
42 1,6,5149541,112389635,1417194907,6,112389635,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389636,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194907,"59000"
43 1,6,5149542,112389637,1417194908,6,112389637,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389638,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194908,"59000"
44 1,6,5149543,112389639,1417194909,6,112389639,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389640,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194909,"59000"
45 1,6,5149544,112389641,1417194910,6,112389641,90334218,"1000","",0,1,4,0,0,0,0,0,0,0,0,0,0,"0","0",112389642,0,0,0,"59000","59000","",0,0,4,0,0,0,0,0,0,0,0,0,0,"0","0",0,1417194910,"59000"
```

Figure 3-2: Section of the Raw CDR data

3.3 Data Preprocessing

Before modeling, the data needed to be processed and presented in the correct format. Data preprocessing is the manipulation of data into a form suitable for further analysis and processing as illustrated in Figure 3-3 below. The main activity during data preprocessing was to select the features to be used for building the neural network model. Each of the call records has a set of call attributes as described earlier. These attributes need to be converted to a form that is compatible with the model. Data pre-processing involved the following:

i. Removing unwanted fields-these are the attributes that were not important in building the model.
ii. Replacing abbreviations with understandable values.
iii. Normalizing values between 0 and 1.
iv. Sorting the data to include only call data with sufficient number calls for modeling.

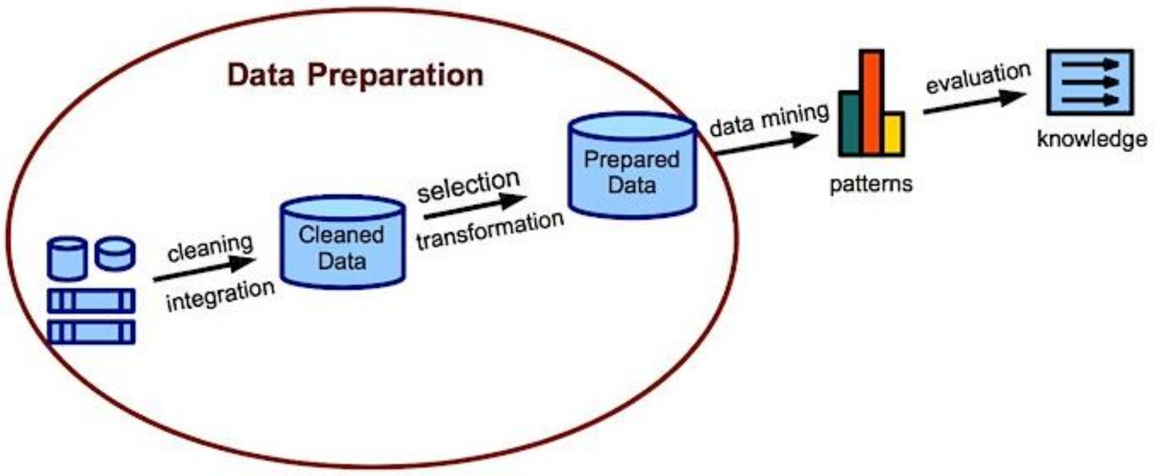


Figure 3-3: Data Preprocessing Phases

3.3.1 Feature Selection

The selection of features strongly influences the subsequent analysis. The goal of feature selection was to capture the desired attributes from the raw CDR data.

The provided CDR records consisted of individual CSV records containing all 104 attributes describing the call made through the system. Some attributes were noted to be of a zero value and were therefore not important in the analysis. On further analysis of the CDR records for the different call scenarios (Transferred calls, Conference calls, abandoned calls, Intercom Calls etc.) and from literature reviewed together with domain knowledge, the following set of attributes in table 3-2 were selected:

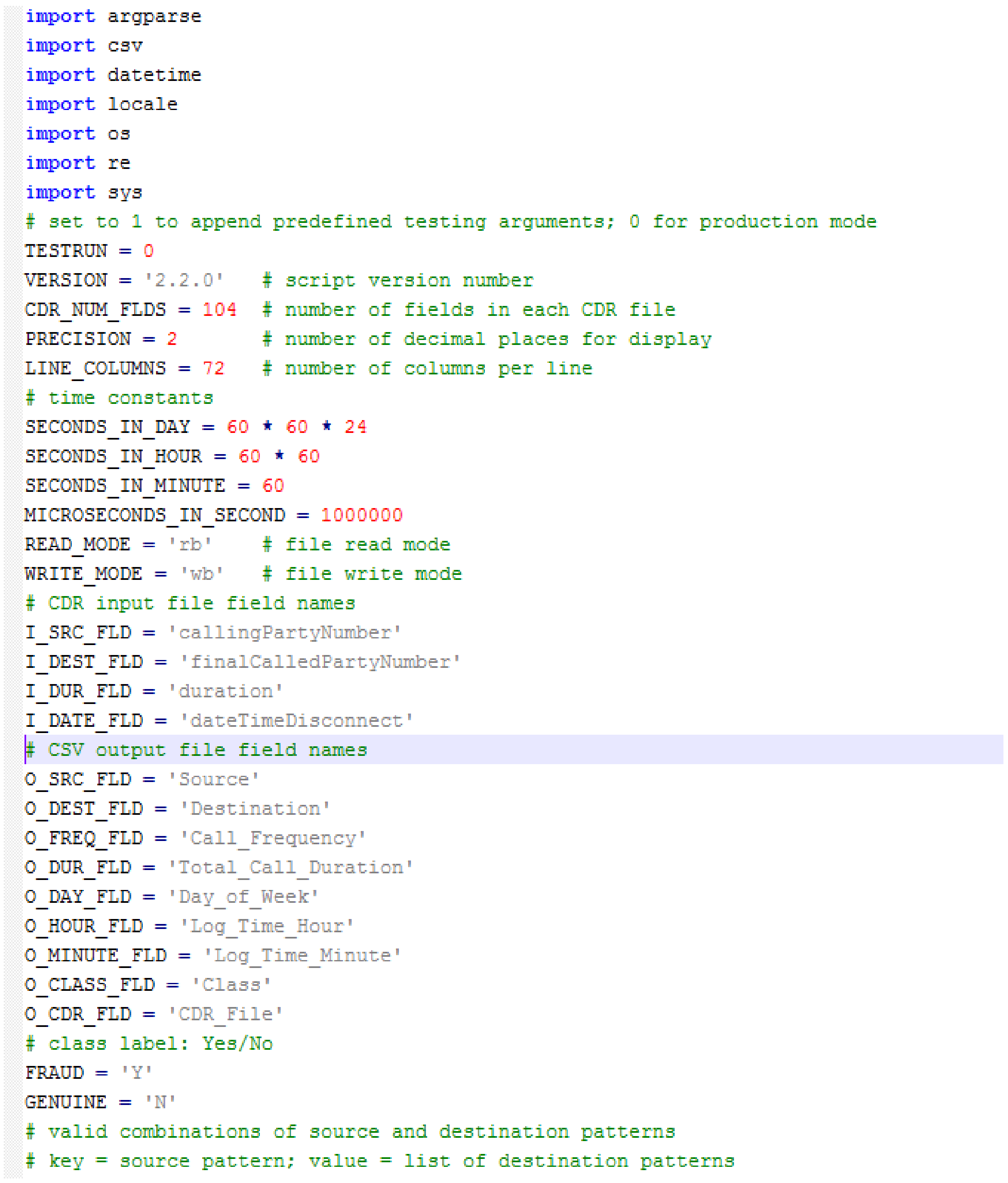
Table 3-2: Feature Selection

Attribute	**Range of Value**	**Description**
Calling Party Number	Text String	This attribute shows the extension number of the line that is used.
Date and Time Connected	0, Integer	This attribute identifies the date and time that the call connects.
Date and Time Disconnected	0, Integer	This attribute identifies the date and time that the call clears.
Destination Device Name	Text String	This field specifies the text string that identifies the name of the destination device.

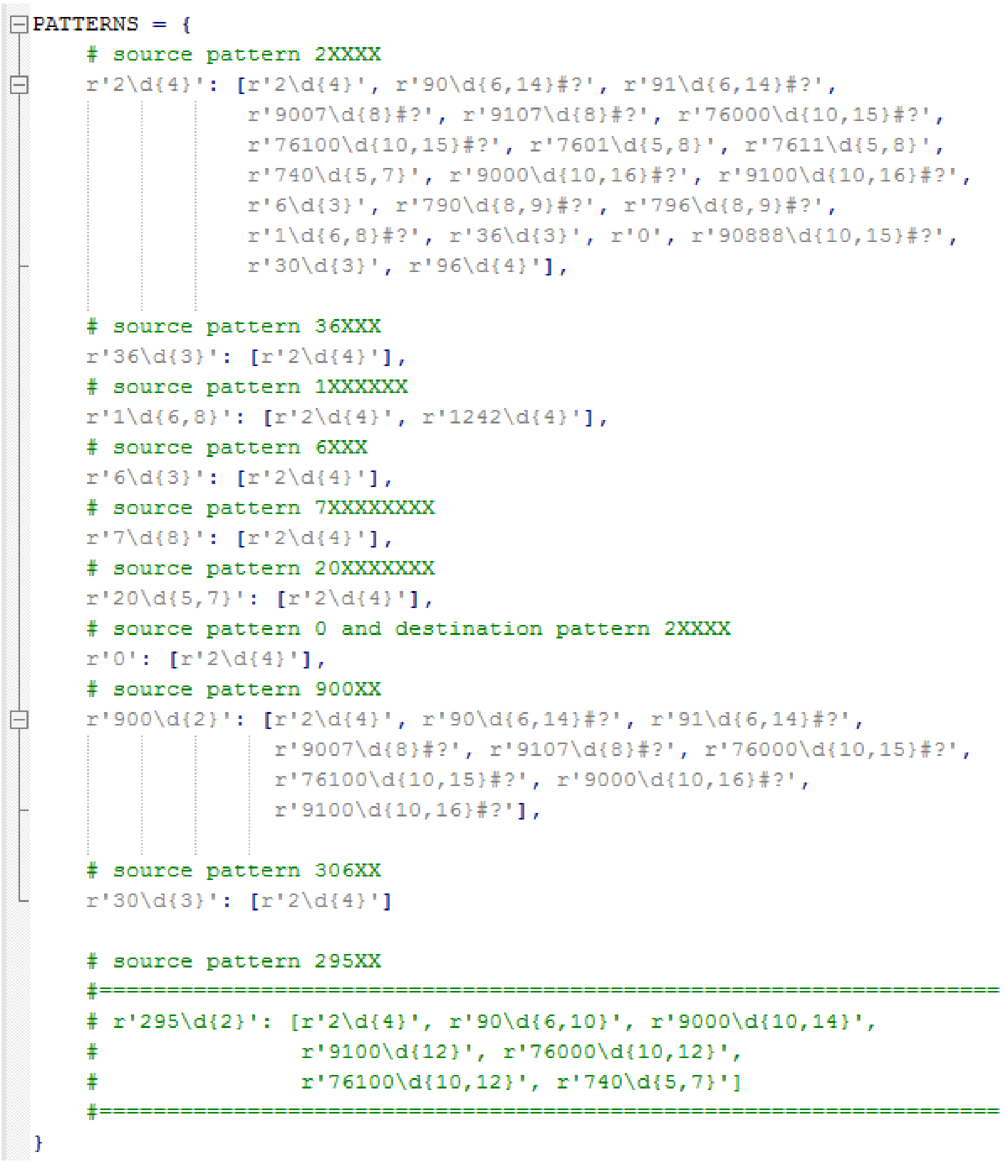
destination Ipv4 Address	Text String	This field identifies the IP address of the device that terminates the call signaling. The field can be either in IPv4 or IPv6 format depending upon the type of IP address that gets used for the call.
Duration	0, Positive integer	This field identifies the difference between the Connect Time and Disconnect Time. This field specifies the time that the call remains connected, in seconds. This field remains zero if the call never connects or if it connects for less than 1 second.
Final Called Party Number	Text String	This field specifies the number to which the call finally gets presented, until it is answered or rings out. If no forwarding occurs, this number shows the same number as the originalCalledPartyNumber.
Originating Device Name	Text String	This field specifies the text string that identifies the name of the originating device
Original Called Party Number	Text String	This field specifies the number to which the original call was presented, prior to any call forwarding.
Originating IP Address	Integer	This field identifies the v4 IP address of the device that originates the call signaling
origin Ipv4v6 Address	Integer	This field identifies the IP address of the device that originates the call signaling. The field can be either IPv4 or IPv6 format depending on the type of IP address that gets used for the call.

3.3.2 Preprocessing Script

A Python data preprocessing script was written to read into the raw CDR data and select the above features described above then label the records as either genuine or fraudulent based on the calling patterns. Figure 3-4 below shows a section of the script.



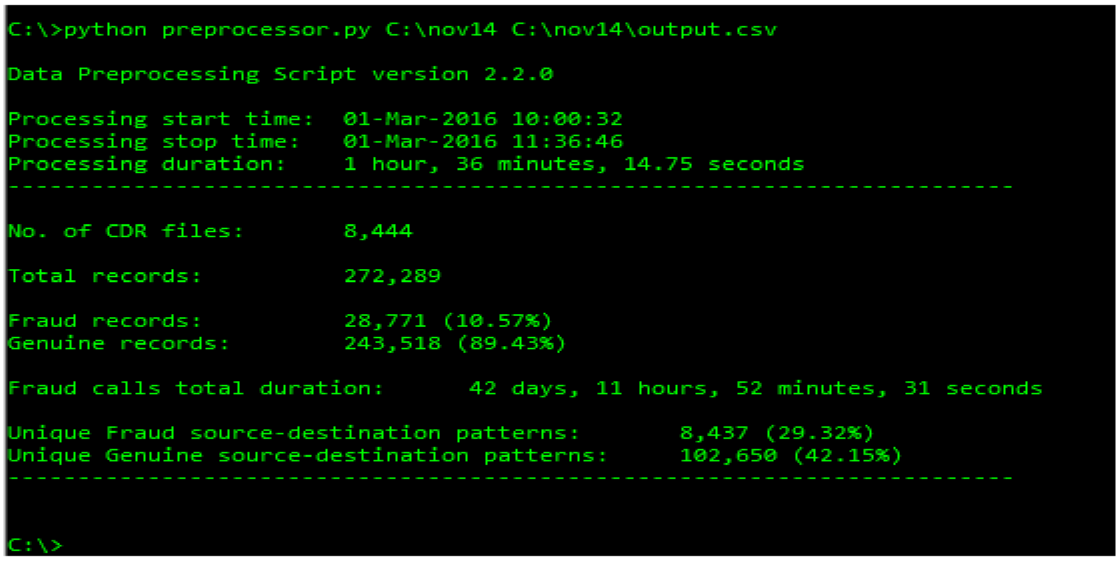
```
import argparse
import csv
import datetime
import locale
import os
import re
import sys
# set to 1 to append predefined testing arguments; 0 for production mode
TESTRUN = 0
VERSION = '2.2.0'   # script version number
CDR_NUM_FLDS = 104  # number of fields in each CDR file
PRECISION = 2       # number of decimal places for display
LINE_COLUMNS = 72   # number of columns per line
# time constants
SECONDS_IN_DAY = 60 * 60 * 24
SECONDS_IN_HOUR = 60 * 60
SECONDS_IN_MINUTE = 60
MICROSECONDS_IN_SECOND = 1000000
READ_MODE = 'rb'    # file read mode
WRITE_MODE = 'wb'   # file write mode
# CDR input file field names
I_SRC_FLD = 'callingPartyNumber'
I_DEST_FLD = 'finalCalledPartyNumber'
I_DUR_FLD = 'duration'
I_DATE_FLD = 'dateTimeDisconnect'
# CSV output file field names
O_SRC_FLD = 'Source'
O_DEST_FLD = 'Destination'
O_FREQ_FLD = 'Call_Frequency'
O_DUR_FLD = 'Total_Call_Duration'
O_DAY_FLD = 'Day_of_Week'
O_HOUR_FLD = 'Log_Time_Hour'
O_MINUTE_FLD = 'Log_Time_Minute'
O_CLASS_FLD = 'Class'
O_CDR_FLD = 'CDR_File'
# class label: Yes/No
FRAUD = 'Y'
GENUINE = 'N'
# valid combinations of source and destination patterns
# key = source pattern; value = list of destination patterns
```



```python
PATTERNS = {
    # source pattern 2XXXX
    r'2\d{4}': [r'2\d{4}', r'90\d{6,14}#?', r'91\d{6,14}#?',
                r'9007\d{8}#?', r'9107\d{8}#?', r'76000\d{10,15}#?',
                r'76100\d{10,15}#?', r'7601\d{5,8}', r'7611\d{5,8}',
                r'740\d{5,7}', r'9000\d{10,16}#?', r'9100\d{10,16}#?',
                r'6\d{3}', r'790\d{8,9}#?', r'796\d{8,9}#?',
                r'1\d{6,8}#?', r'36\d{3}', r'0', r'90888\d{10,15}#?',
                r'30\d{3}', r'96\d{4}'],

    # source pattern 36XXX
    r'36\d{3}': [r'2\d{4}'],
    # source pattern 1XXXXXX
    r'1\d{6,8}': [r'2\d{4}', r'1242\d{4}'],
    # source pattern 6XXX
    r'6\d{3}': [r'2\d{4}'],
    # source pattern 7XXXXXXXX
    r'7\d{8}': [r'2\d{4}'],
    # source pattern 20XXXXXXX
    r'20\d{5,7}': [r'2\d{4}'],
    # source pattern 0 and destination pattern 2XXXX
    r'0': [r'2\d{4}'],
    # source pattern 900XX
    r'900\d{2}': [r'2\d{4}', r'90\d{6,14}#?', r'91\d{6,14}#?',
                  r'9007\d{8}#?', r'9107\d{8}#?', r'76000\d{10,15}#?',
                  r'76100\d{10,15}#?', r'9000\d{10,16}#?',
                  r'9100\d{10,16}#?'],

    # source pattern 306XX
    r'30\d{3}': [r'2\d{4}']

    # source pattern 295XX
    #====================================================================
    # r'295\d{2}': [r'2\d{4}', r'90\d{6,10}', r'9000\d{10,14}',
    #               r'9100\d{12}', r'76000\d{10,12}',
    #               r'76100\d{10,12}', r'740\d{5,7}']
    #====================================================================
}
```

Figure 3-4: Section of the Python Preprocessing Script

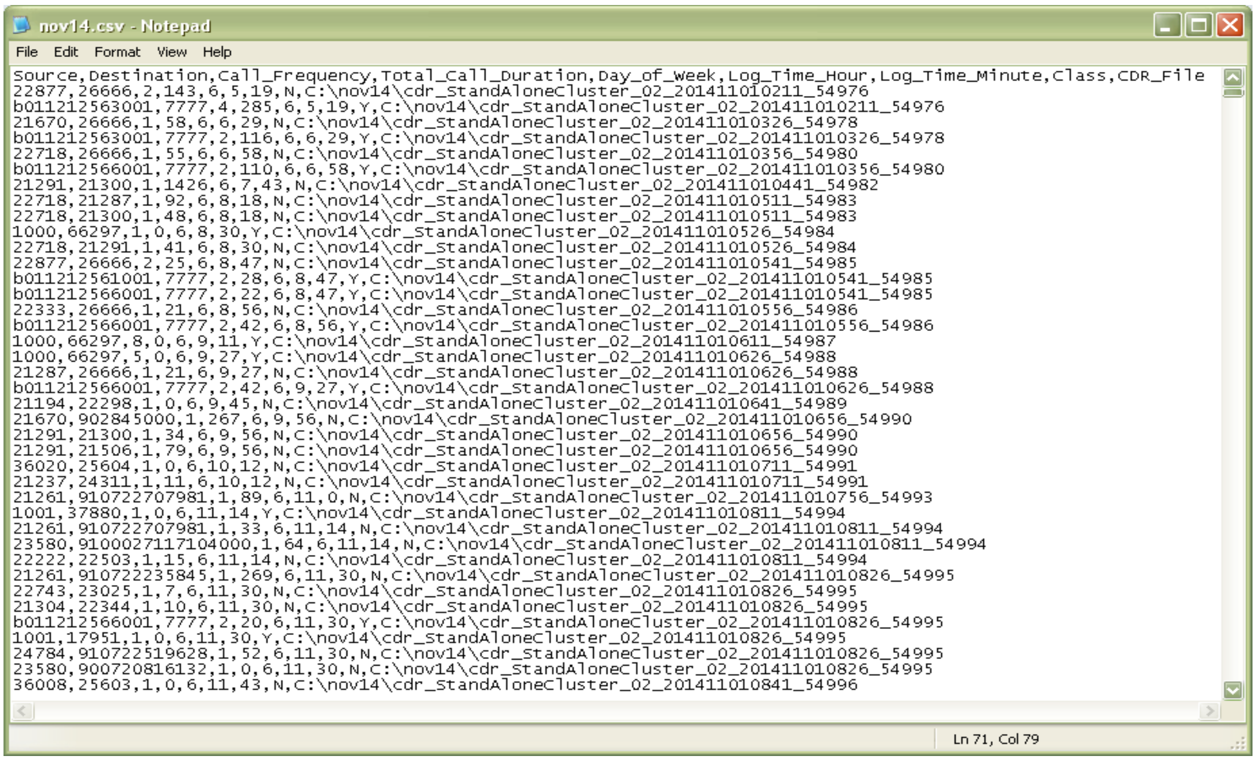
Figure 3-5 below shows a sample output of the python data preprocessing script on the raw CDR data. The script was written to read into the individual CDR files and then output a csv file that contains only attributes important in determining fraud patterns as described in the feature selection section.



```
C:\>python preprocessor.py C:\nov14 C:\nov14\output.csv

Data Preprocessing Script version 2.2.0

Processing start time:  01-Mar-2016 10:00:32
Processing stop time:   01-Mar-2016 11:36:46
Processing duration:    1 hour, 36 minutes, 14.75 seconds
--------------------------------------------------------------------------

No. of CDR files:       8,444

Total records:          272,289

Fraud records:          28,771 (10.57%)
Genuine records:        243,518 (89.43%)

Fraud calls total duration:      42 days, 11 hours, 52 minutes, 31 seconds

Unique Fraud source-destination patterns:       8,437 (29.32%)
Unique Genuine source-destination patterns:     102,650 (42.15%)
--------------------------------------------------------------------------


C:\>
```

Figure 3-5: Data preprocessing script output

Figure 3-6 shows a section of a preprocessed output file containing the selected features only-originating number of the call, destination number, the frequency of the call within the individual CDR file, duration of the call-these attributes are important in determining fraudulent behavior..



```
Source,Destination,Call_Frequency,Total_Call_Duration,Day_of_Week,Log_Time_Hour,Log_Time_Minute,Class,CDR_File
22877,26666,2,143,6,5,19,N,C:\nov14\cdr_StandAloneCluster_02_201411010211_54976
b011212563001,7777,4,285,6,5,19,Y,C:\nov14\cdr_StandAloneCluster_02_201411010211_54976
21670,26666,1,58,6,6,29,N,C:\nov14\cdr_StandAloneCluster_02_201411010326_54978
b011212563001,7777,2,116,6,6,29,Y,C:\nov14\cdr_StandAloneCluster_02_201411010326_54978
22718,26666,1,55,6,6,58,N,C:\nov14\cdr_StandAloneCluster_02_201411010356_54980
b011212566001,7777,2,110,6,6,58,Y,C:\nov14\cdr_StandAloneCluster_02_201411010356_54980
21291,21300,1,1426,6,7,43,N,C:\nov14\cdr_StandAloneCluster_02_201411010441_54982
22718,21287,1,92,6,8,18,N,C:\nov14\cdr_StandAloneCluster_02_201411010511_54983
22718,21300,1,48,6,8,18,N,C:\nov14\cdr_StandAloneCluster_02_201411010511_54983
1000,66297,1,0,6,8,30,Y,C:\nov14\cdr_StandAloneCluster_02_201411010526_54984
22718,21291,1,41,6,8,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010526_54984
22877,26666,2,25,6,8,47,N,C:\nov14\cdr_StandAloneCluster_02_201411010541_54985
b011212561001,7777,2,28,6,8,47,Y,C:\nov14\cdr_StandAloneCluster_02_201411010541_54985
b011212566001,7777,2,22,6,8,47,Y,C:\nov14\cdr_StandAloneCluster_02_201411010541_54985
22333,26666,1,21,6,8,56,N,C:\nov14\cdr_StandAloneCluster_02_201411010556_54986
b011212566001,7777,2,42,6,8,56,Y,C:\nov14\cdr_StandAloneCluster_02_201411010556_54986
1000,66297,8,0,6,9,11,Y,C:\nov14\cdr_StandAloneCluster_02_201411010611_54987
1000,66297,5,0,6,9,27,Y,C:\nov14\cdr_StandAloneCluster_02_201411010626_54988
21287,26666,1,21,6,9,27,N,C:\nov14\cdr_StandAloneCluster_02_201411010626_54988
b011212566001,7777,2,42,6,9,27,Y,C:\nov14\cdr_StandAloneCluster_02_201411010626_54988
21194,22298,1,0,6,9,45,N,C:\nov14\cdr_StandAloneCluster_02_201411010641_54989
21670,902845000,1,267,6,9,56,N,C:\nov14\cdr_StandAloneCluster_02_201411010656_54990
21291,21300,1,34,6,9,56,N,C:\nov14\cdr_StandAloneCluster_02_201411010656_54990
21291,21506,1,79,6,9,56,N,C:\nov14\cdr_StandAloneCluster_02_201411010656_54990
36020,25604,1,0,6,10,12,N,C:\nov14\cdr_StandAloneCluster_02_201411010711_54991
21237,24311,1,11,6,10,12,N,C:\nov14\cdr_StandAloneCluster_02_201411010711_54991
21261,910722707981,1,89,6,11,0,N,C:\nov14\cdr_StandAloneCluster_02_201411010756_54993
1001,37880,1,0,6,11,14,Y,C:\nov14\cdr_StandAloneCluster_02_201411010811_54994
21261,910722707981,1,33,6,11,14,N,C:\nov14\cdr_StandAloneCluster_02_201411010811_54994
23580,9100027117104000,1,64,6,11,14,N,C:\nov14\cdr_StandAloneCluster_02_201411010811_54994
22222,22503,1,15,6,11,14,N,C:\nov14\cdr_StandAloneCluster_02_201411010811_54994
21261,910722235845,1,269,6,11,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
22743,23025,1,7,6,11,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
21304,22344,1,10,6,11,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
b011212566001,7777,2,20,6,11,30,Y,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
1001,17951,1,0,6,11,30,Y,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
24784,910722519628,1,52,6,11,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
23580,900720816132,1,0,6,11,30,N,C:\nov14\cdr_StandAloneCluster_02_201411010826_54995
36008,25603,1,0,6,11,43,N,C:\nov14\cdr_StandAloneCluster_02_201411010841_54996
```

Figure 3-6: Section of a preprocessed output file

Three preprocessed output files were produced in CSV format for each of the three months used in this study. The output files contained the fields specified in the table 3-3 below:

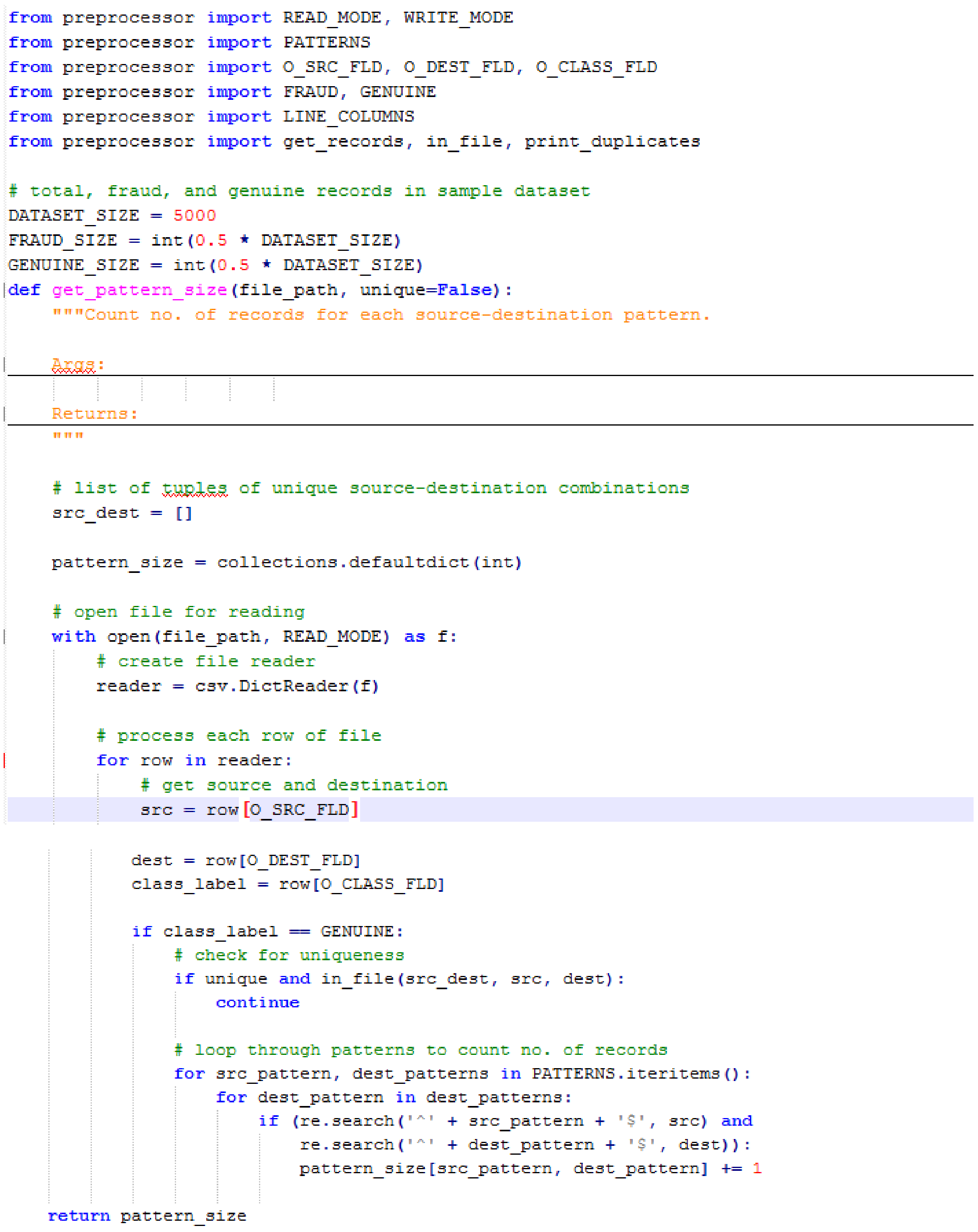
Table 3-3: Attributes of preprocessed data

Attribute	**Description**
Source	This attribute shows the extension number of the line that made the call.
Destination	This field specifies the number to which the call finally gets presented
Call Frequency	This is the number of calls made from a particular source to a particular destination within the specified CDR file
Total Call Duration	This is the total duration (in seconds) of calls made from a particular source to a particular destination within the specified CDR file
Day of Week	This is the day of the week when the CDR file was logged
Time of Day	This is the time of day the CDR file was logged
Class Label	This is the class label that identifies the record as genuine or fraudulent based on the calling patterns.

3.3.3 Sampling Final Data Set

The three preprocessed files produced for the three months used in this study were subjected to random sampling in order to produce a final data set of 15,000 records for building the network model. 5,000 records (2,500 fraud and 2,500 genuine) were sampled from each month's preprocessed file to produce the total of 15,000 records.

In the final data set 50% (7,500) of the records were fraudulent while 50% (7,500) were genuine. An equal percentage was used in order to train the network with an equal number of records of both classes. A section of the python script used to perform the random sampling is shown in figure 3-7 below.

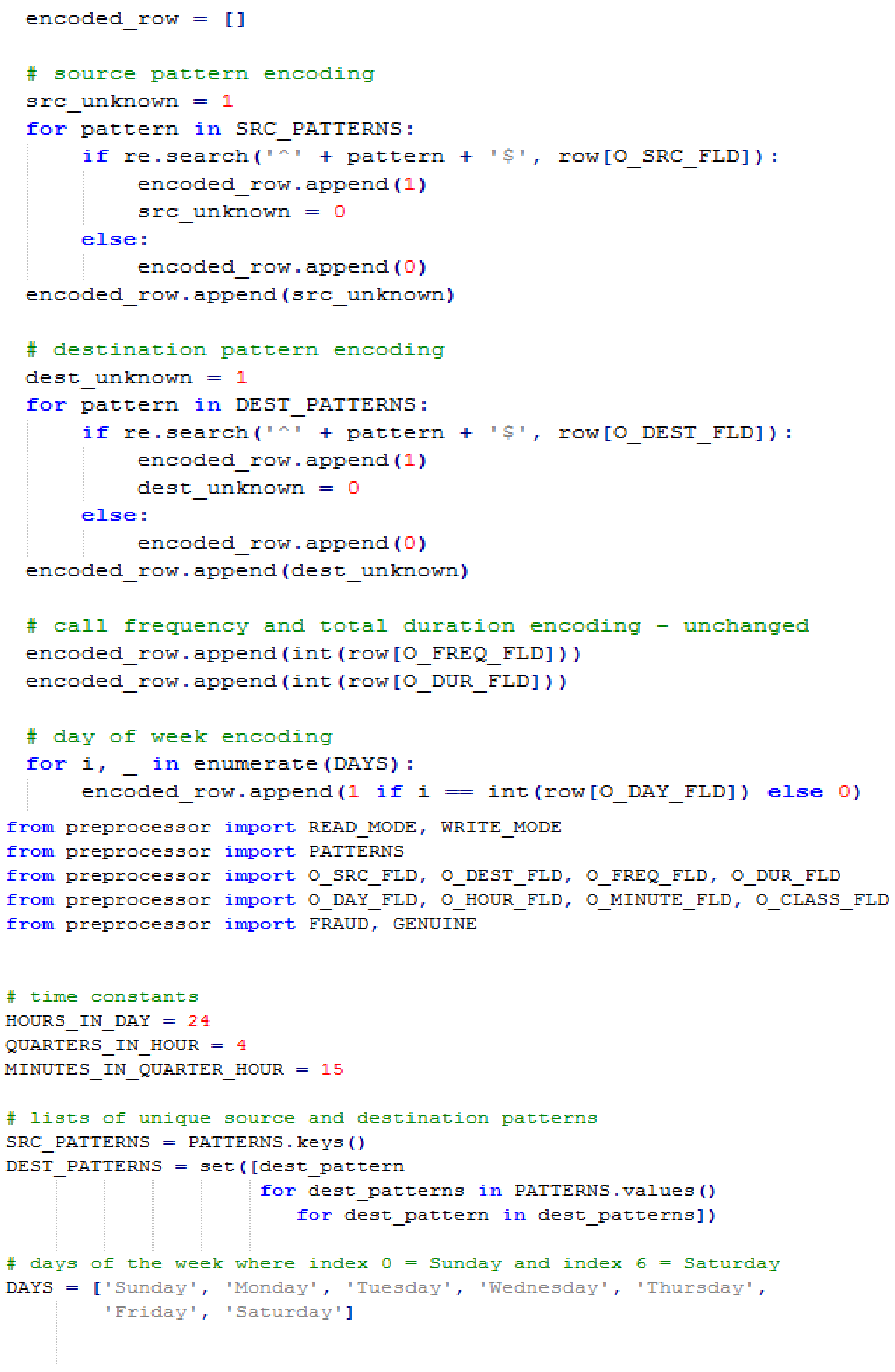


```python
from preprocessor import READ_MODE, WRITE_MODE
from preprocessor import PATTERNS
from preprocessor import O_SRC_FLD, O_DEST_FLD, O_CLASS_FLD
from preprocessor import FRAUD, GENUINE
from preprocessor import LINE_COLUMNS
from preprocessor import get_records, in_file, print_duplicates

# total, fraud, and genuine records in sample dataset
DATASET_SIZE = 5000
FRAUD_SIZE = int(0.5 * DATASET_SIZE)
GENUINE_SIZE = int(0.5 * DATASET_SIZE)
def get_pattern_size(file_path, unique=False):
    """Count no. of records for each source-destination pattern.

    Args:

    Returns:
    """

    # list of tuples of unique source-destination combinations
    src_dest = []

    pattern_size = collections.defaultdict(int)

    # open file for reading
    with open(file_path, READ_MODE) as f:
        # create file reader
        reader = csv.DictReader(f)

        # process each row of file
        for row in reader:
            # get source and destination
            src = row[O_SRC_FLD]

            dest = row[O_DEST_FLD]
            class_label = row[O_CLASS_FLD]

            if class_label == GENUINE:
                # check for uniqueness
                if unique and in_file(src_dest, src, dest):
                    continue

                # loop through patterns to count no. of records
                for src_pattern, dest_patterns in PATTERNS.iteritems():
                    for dest_pattern in dest_patterns:
                        if (re.search('^' + src_pattern + '$', src) and
                            re.search('^' + dest_pattern + '$', dest)):
                            pattern_size[src_pattern, dest_pattern] += 1

    return pattern_size
```

Figure 3-7: Section of the python Script for random sampling

3.3.4 Encoding in Matlab Format

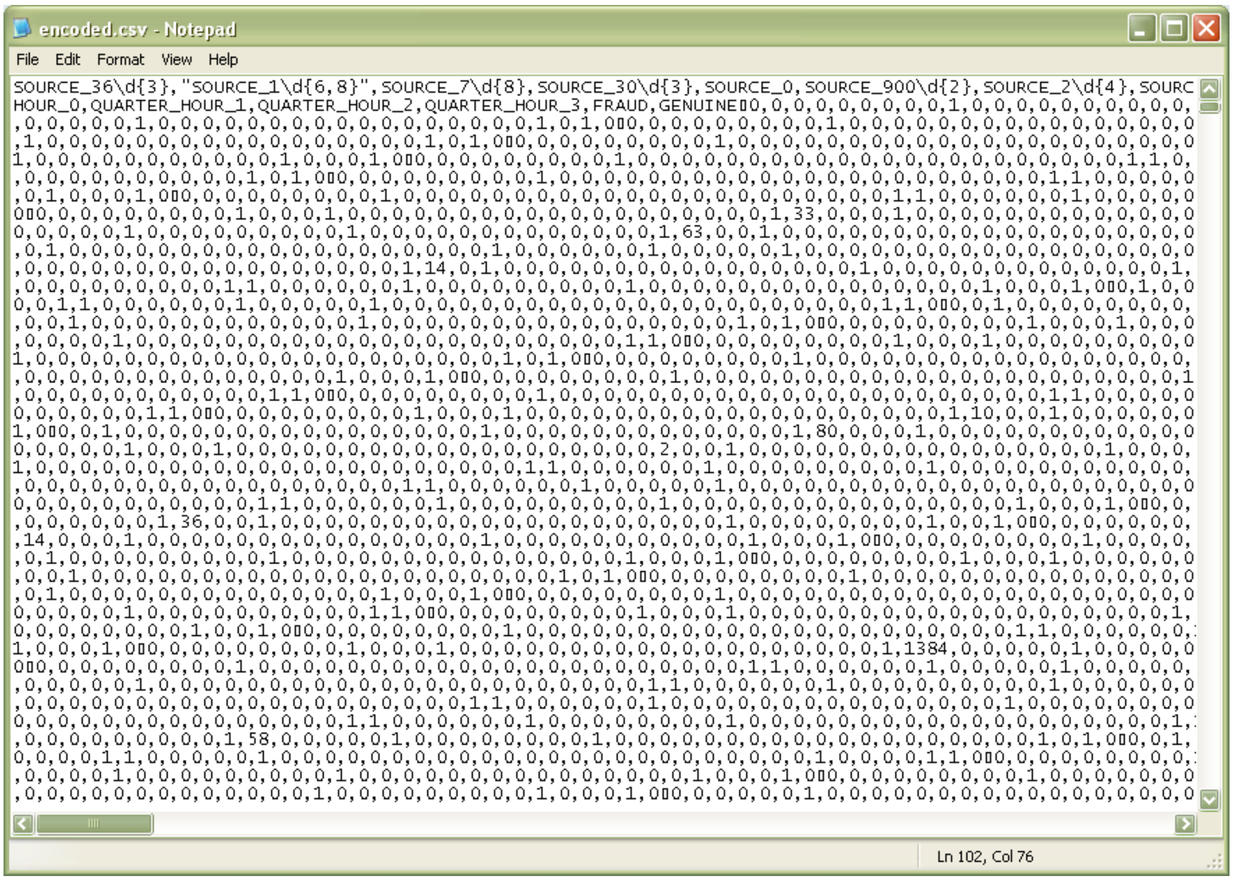
The sampled final data set of 15,000 records was encoded in a binary 1-of-N encoding format for use by the neural network built using the MATLAB neural network toolbox. A section of the python script written to perform this binary encoding is shown in figure 3-8 below.



```
  encoded_row = []

  # source pattern encoding
  src_unknown = 1
  for pattern in SRC_PATTERNS:
      if re.search('^' + pattern + '$', row[O_SRC_FLD]):
          encoded_row.append(1)
          src_unknown = 0
      else:
          encoded_row.append(0)
  encoded_row.append(src_unknown)

  # destination pattern encoding
  dest_unknown = 1
  for pattern in DEST_PATTERNS:
      if re.search('^' + pattern + '$', row[O_DEST_FLD]):
          encoded_row.append(1)
          dest_unknown = 0
      else:
          encoded_row.append(0)
  encoded_row.append(dest_unknown)

  # call frequency and total duration encoding - unchanged
  encoded_row.append(int(row[O_FREQ_FLD]))
  encoded_row.append(int(row[O_DUR_FLD]))

  # day of week encoding
  for i, _ in enumerate(DAYS):
      encoded_row.append(1 if i == int(row[O_DAY_FLD]) else 0)
from preprocessor import READ_MODE, WRITE_MODE
from preprocessor import PATTERNS
from preprocessor import O_SRC_FLD, O_DEST_FLD, O_FREQ_FLD, O_DUR_FLD
from preprocessor import O_DAY_FLD, O_HOUR_FLD, O_MINUTE_FLD, O_CLASS_FLD
from preprocessor import FRAUD, GENUINE


# time constants
HOURS_IN_DAY = 24
QUARTERS_IN_HOUR = 4
MINUTES_IN_QUARTER_HOUR = 15

# lists of unique source and destination patterns
SRC_PATTERNS = PATTERNS.keys()
DEST_PATTERNS = set([dest_pattern
                     for dest_patterns in PATTERNS.values()
                        for dest_pattern in dest_patterns])

# days of the week where index 0 = Sunday and index 6 = Saturday
DAYS = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday',
        'Friday', 'Saturday']
```

Figure 3-8: A section Python script for binary encoding

The encoding script produced a CSV file with 70 fields of data attributes and 2 fields of class labels (fraud and genuine). The 70 fields result from creating 31 fields for the genuine source and destination patterns, 2 fields for unknown sources and destinations, 2 fields for call frequency and total call duration, 7 fields for each day of the week, 24 fields for each hour in a day and 4 fields for each quarter-hour period in an hour. A section of the encoded data is shown figure 3-9 below



```
SOURCE_36\d{3},"SOURCE_1\d{6,8}",SOURCE_7\d{8},SOURCE_30\d{3},SOURCE_0,SOURCE_900\d{2},SOURCE_2\d{4},SOURC
HOUR_0,QUARTER_HOUR_1,QUARTER_HOUR_2,QUARTER_HOUR_3,FRAUD,GENUINE▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,
,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0
,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0
0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,33,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,63,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,14,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,
,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,1,0,0
0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0▯0,0,1,0,0,0,0,0,0,0,0,0,
,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
,0,0,0,0,0,0,0,0,0,0,0,1,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0
0,0,0,0,0,0,1,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,10,0,0,1,0,0,0,0,0
1,0▯0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,80,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,
,0,0,0,0,0,0,1,36,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,1,0▯0,0,0,0,0,0,0,0,
,14,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,
,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0
,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,:
1,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1384,0,0,0,0,0,1,0,0,0,0,0
0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,
,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,:
,0,0,0,0,0,0,0,0,0,0,1,58,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0▯0,0,1,
0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0▯0,0,0,0,0,0,0,0,:
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0▯0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0▯0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Ln 102, Col 76

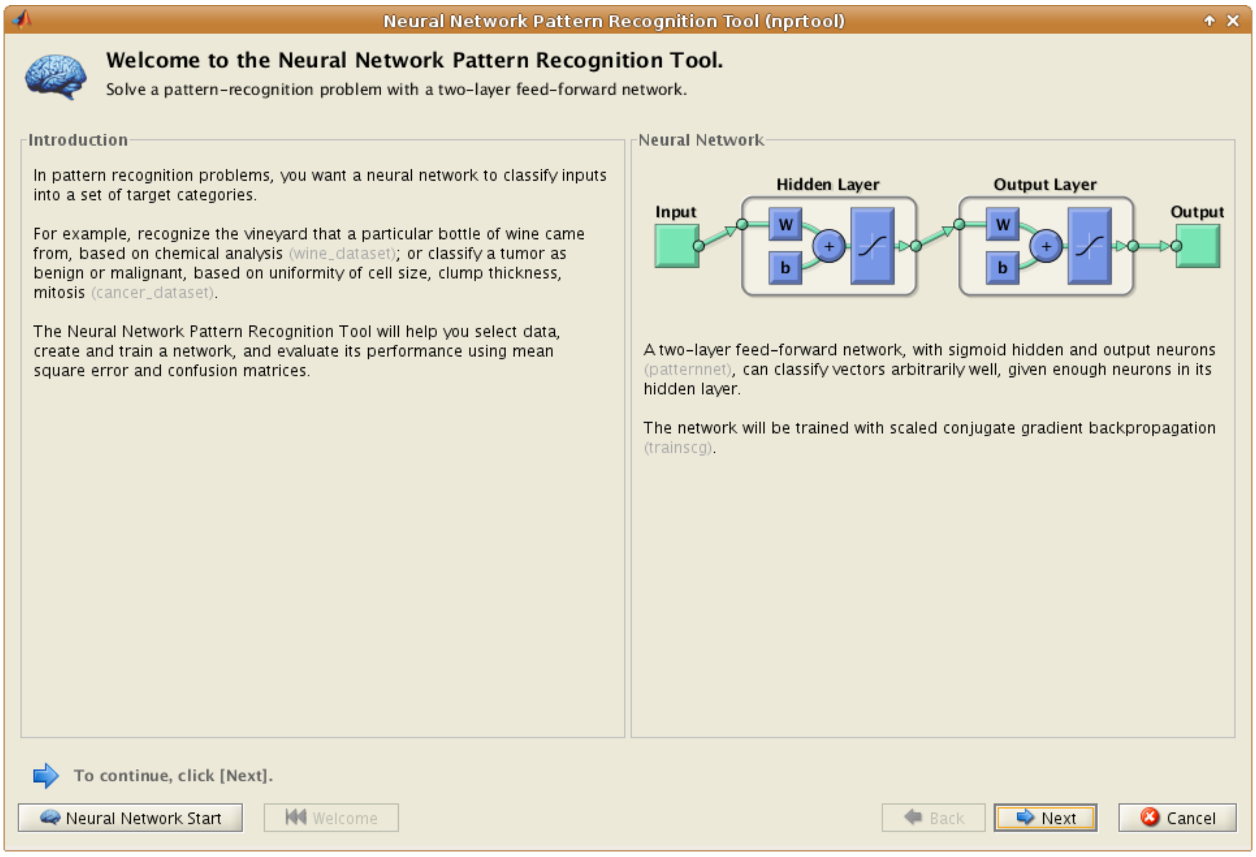
Figure 3-9: Section of the encoded final data set

The Python script also produced a MATLAB data file from the encoded CSV file. This was the data set used by the MATLAB neural network toolbox. The MATLAB data file contained two matrices:

i. The 15,000 X 70 **inputs** matrix contained 70 data fields for the 15,000 sample records
ii. The 15,000 X 2 **targets** matrix contained 2 class label fields for the 15,000 sample records

3.4 Network Building

The MATLAB neural network toolbox was used to design the network, train it and test it on the sample data set as shown in Figure 3-10 below. The Neural Network Pattern Recognition Tool was used for this task.


Neural Network Pattern Recognition Tool (nprtool)

Welcome to the Neural Network Pattern Recognition Tool.

Solve a pattern-recognition problem with a two-layer feed-forward network.

Introduction

In pattern recognition problems, you want a neural network to classify inputs into a set of target categories.

For example, recognize the vineyard that a particular bottle of wine came from, based on chemical analysis (wine_dataset); or classify a tumor as benign or malignant, based on uniformity of cell size, clump thickness, mitosis (cancer_dataset).

The Neural Network Pattern Recognition Tool will help you select data, create and train a network, and evaluate its performance using mean square error and confusion matrices.

Neural Network

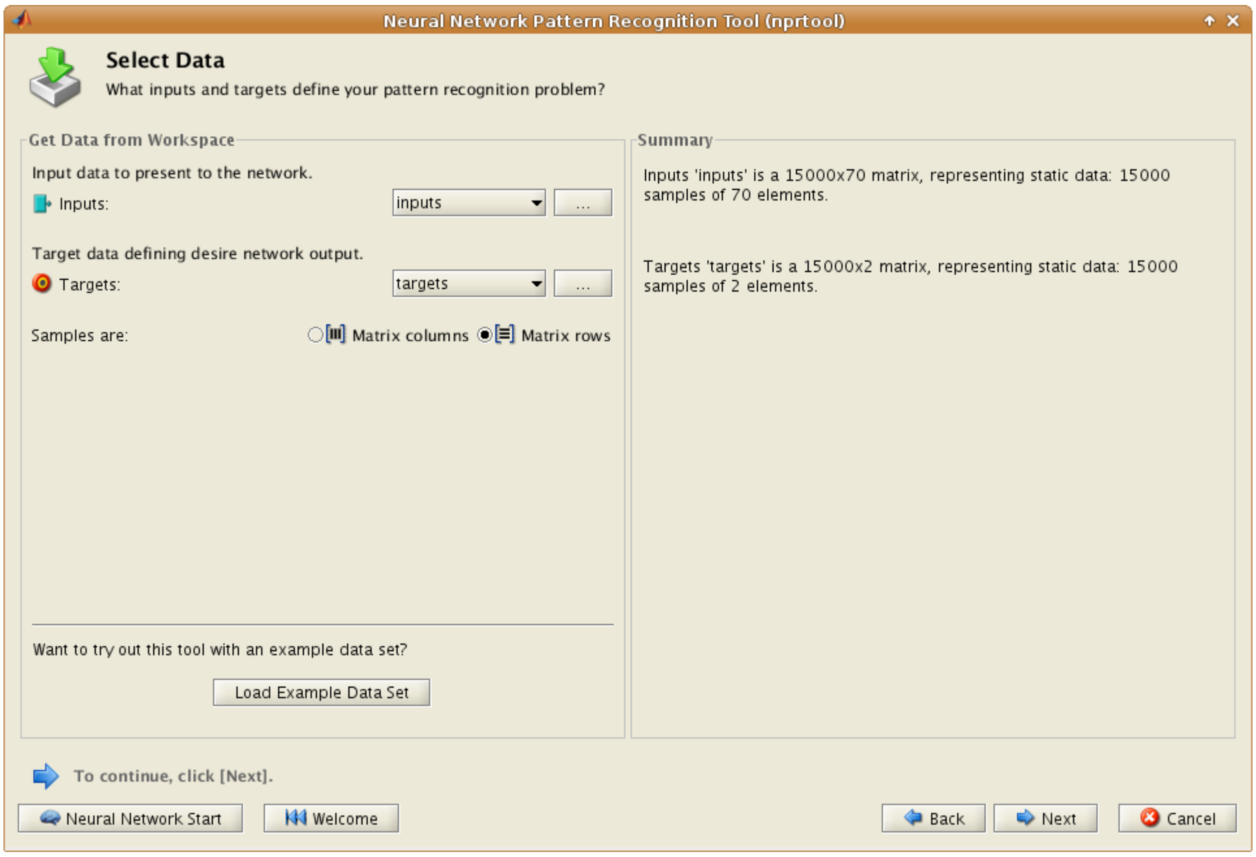
A two-layer feed-forward network, with sigmoid hidden and output neurons (patternnet), can classify vectors arbitrarily well, given enough neurons in its hidden layer.

The network will be trained with scaled conjugate gradient backpropagation (trainscg).

To continue, click [Next].

Figure 3-10: The Neural Network Pattern Recognition Tool

The first step was to select the sample data set created during the data preprocessing phase. The inputs and targets were defined at this step as shown in Figure 3-11 below.


Neural Network Pattern Recognition Tool (nprtool)
Select Data
What inputs and targets define your pattern recognition problem?
Get Data from Workspace
Input data to present to the network.
Inputs:
inputs
Target data defining desire network output.
Targets:
targets
Samples are:
Matrix columns
Matrix rows
Want to try out this tool with an example data set?
Load Example Data Set
Summary
To continue, click [Next].
Neural Network Start
Welcome
Back
Next
Cancel

Samples are: ○ Matrix columns [✓ ● Matrix rows]

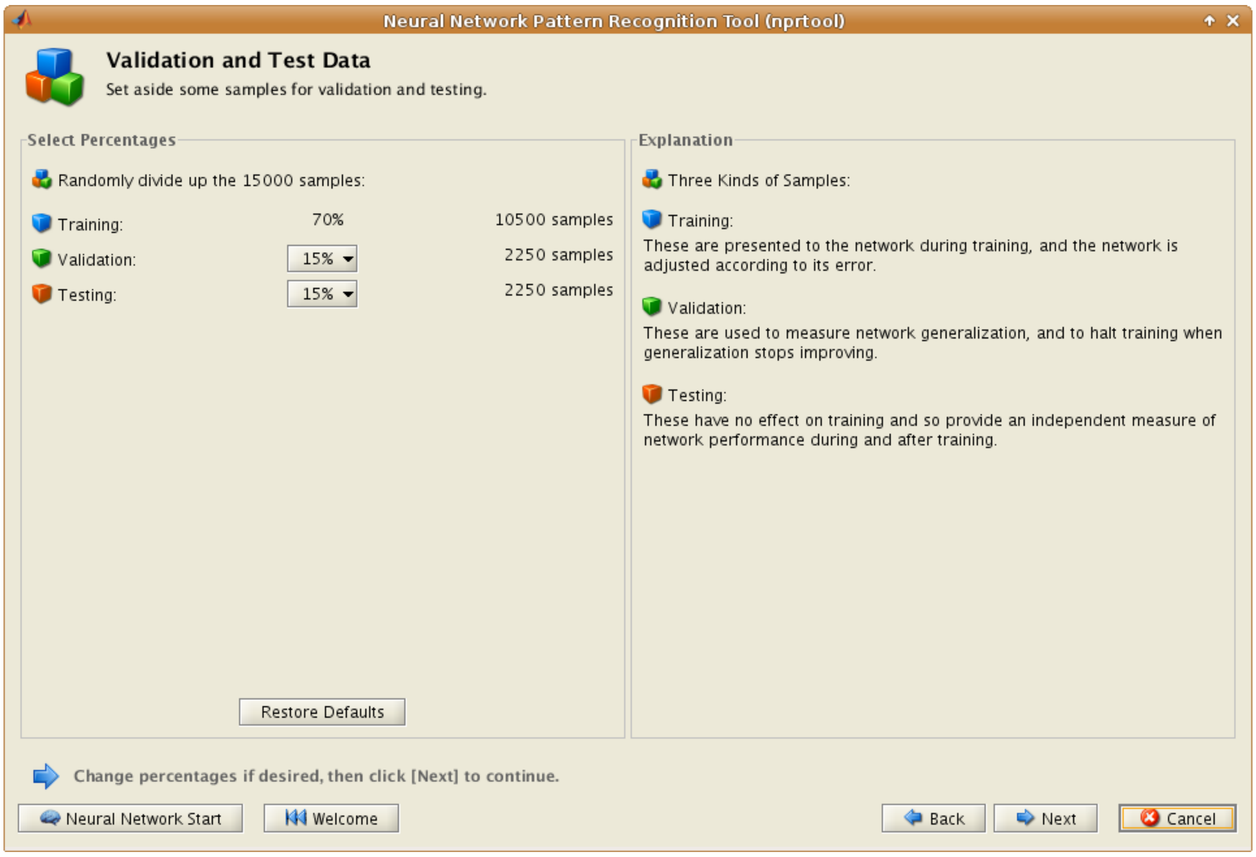
Inputs 'inputs' is a 15000x70 matrix, representing static data: 15000 samples of 70 elements.

Targets 'targets' is a 15000x2 matrix, representing static data: 15000 samples of 2 elements.

Figure 3-11: Data selection

The next step was to partition the 15,000 sample records into three sets: training set, validation set and test set. The training set was used in training the network. The validation set was used to determine when to stop the training. Finally, the test set was used in testing the accuracy of the model.

In this study 70% of the data set (10,500 records) was set aside for training, 15% (2,250 records) for validation and the remaining 15% (2,250 records) was the test set as shown in Figure 3-12 below. This partitioning was done randomly by the Matlab Neural Network Pattern Recognition Tool.


Neural Network Pattern Recognition Tool (nprtool)

Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 15000 samples:

Training:	70%	10500 samples
Validation:	15%	2250 samples
Testing:	15%	2250 samples

Explanation

Three Kinds of Samples:

Training:
These are presented to the network during training, and the network is adjusted according to its error.

Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.

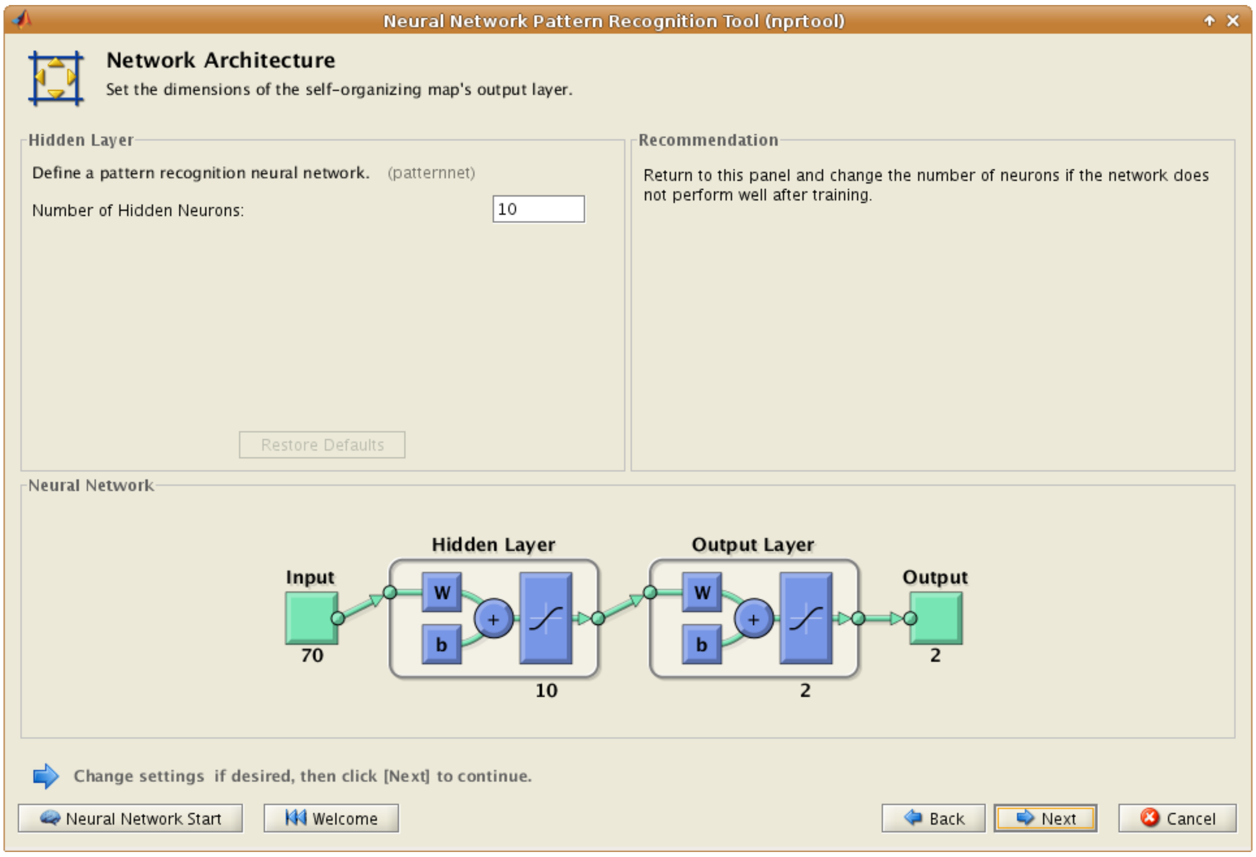
Change percentages if desired, then click [Next] to continue.

Figure 3-12: Data partitioning

The next step was designing the architecture of the neural network. The network had three layers:

i. The input layer of 70 data attributes from the sample data set
ii. Hidden layer containing 10 neurons
iii. The output layer containing 2 neurons for the 2 class labels in the data set (fraud and genuine)

Figure 3-13 below shows the resulting network architecture. The standard network that is used for pattern recognition in the Matlab Neural Network Toolbox is a two-layer feedforward network, with sigmoid transfer functions in both the hidden layer and the output layer.


Neural Network Pattern Recognition Tool (nprtool)

Network Architecture

Set the dimensions of the self-organizing map's output layer.

Hidden Layer

Define a pattern recognition neural network. (patternnet)

Number of Hidden Neurons: 10

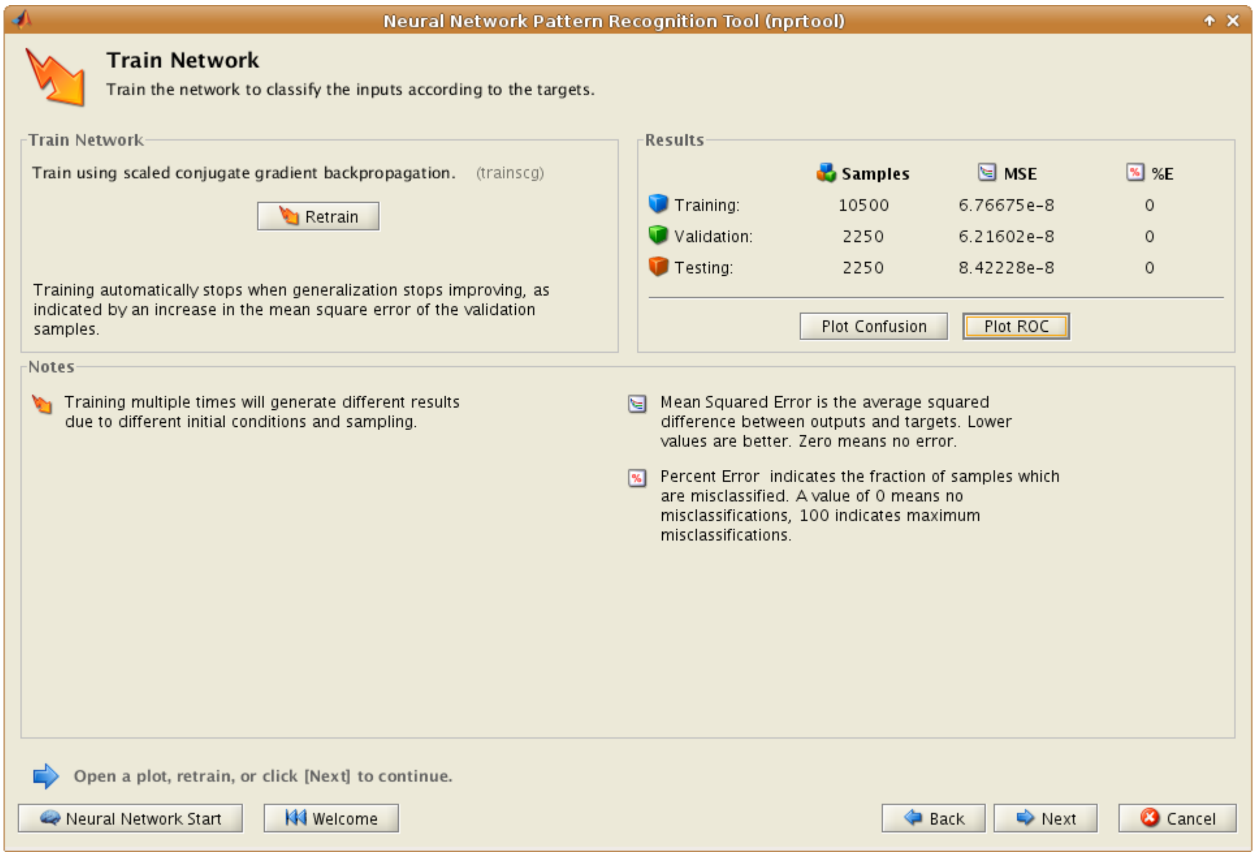
Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Change settings if desired, then click [Next] to continue.

Figure 3-13: Network architecture

After designing the neural network architecture, the next step was training the network to classify the inputs according to the targets as shown in figure 3-14 below. The network was trained using scaled conjugate gradient back propagation.



Neural Network Pattern Recognition Tool (nprtool)

Train Network

Train the network to classify the inputs according to the targets.

Train Network

Train using scaled conjugate gradient backpropagation. (trainscg)

Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Results

	Samples	MSE	%E
Training:	10500	6.76675e-8	0
Validation:	2250	6.21602e-8	0
Testing:	2250	8.42228e-8	0

Notes

Training multiple times will generate different results due to different initial conditions and sampling.

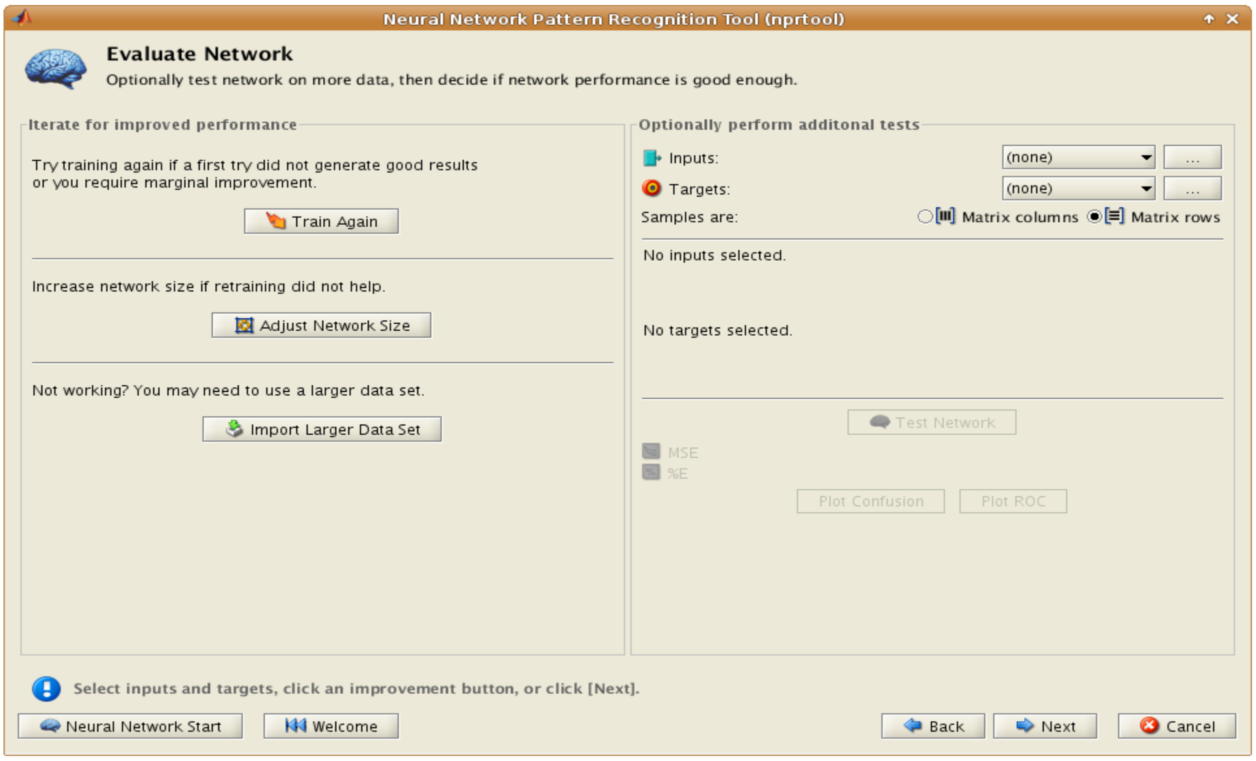
Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Percent Error indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.

Open a plot, retrain, or click [Next] to continue.

Figure 3-14: Training the network

After training the network the network was tested with more data, the next step was evaluating its performance as shown in figure 3-15 below.


Neural Network Pattern Recognition Tool (nprtool)

Evaluate Network
Optionally test network on more data, then decide if network performance is good enough.

Iterate for improved performance

Try training again if a first try did not generate good results or you require marginal improvement.

Train Again

Increase network size if retraining did not help.

Adjust Network Size

Not working? You may need to use a larger data set.

Import Larger Data Set

Optionally perform additonal tests

Inputs: (none)
Targets: (none)
Samples are: Matrix columns [● Matrix rows]

No inputs selected.

No targets selected.

Test Network

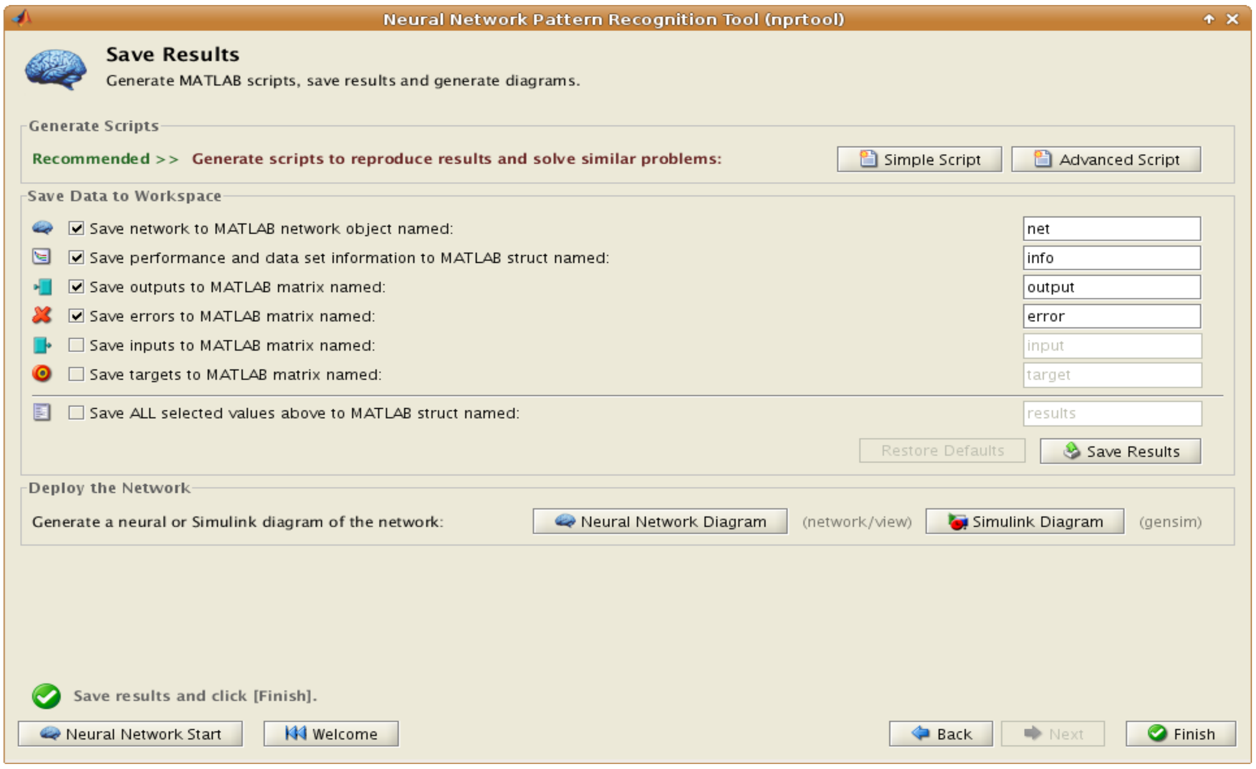
MSE
%E

Plot Confusion | Plot ROC

Select inputs and targets, click an improvement button, or click [Next].

Figure 3-15: Evaluating the network

The final step was saving the Matlab network object together with the training results as shown in figure 3-16 below.


Neural Network Pattern Recognition Tool (nprtool)

Save Results
Generate MATLAB scripts, save results and generate diagrams.

Generate Scripts

Recommended >> Generate scripts to reproduce results and solve similar problems:

Simple Script | Advanced Script

Save Data to Workspace

✓ Save network to MATLAB network object named:	net
✓ Save performance and data set information to MATLAB struct named:	info
✓ Save outputs to MATLAB matrix named:	output
✓ Save errors to MATLAB matrix named:	error
Save inputs to MATLAB matrix named:	input
Save targets to MATLAB matrix named:	target
Save ALL selected values above to MATLAB struct named:	results

Restore Defaults | Save Results

Deploy the Network

Generate a neural or Simulink diagram of the network: Neural Network Diagram (network/view) Simulink Diagram (gensim)

Save results and click [Finish].

Figure 3-16: Save results

3.5 Web Application Development

A web application for reading and classifying the call records in new CDR files was developed as part of this study. It utilized the trained neural network and provided a visualization of the classified records. The application was developed using the agile system development methodology as per the activities highlighted in Figure 3-17 below. The activities carried out during system development were: Requirements Gathering, System Design, Coding, Testing, and Deployment.

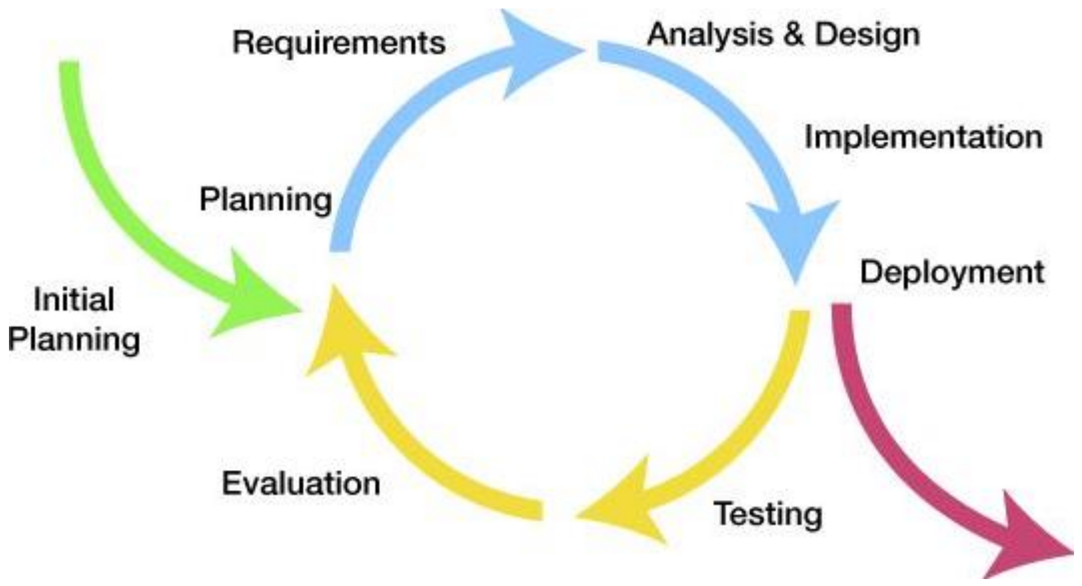


Figure 3-17: Agile system development methodology

3.5.1 Requirements Gathering

The functional requirements for the web application were gathered through structured interviews with administrator of VoIP systems. After the requirements had been gathered, they were used to inform the system analysis phase. Table 3-4 summarizes the functional requirements. It lists software attributes and the functional requirements for these attributes.

Table 3-4: Web application functional requirements

Attribute	**Functional Requirement**
Accessibility	The web application shall be accessible to all relevant staff over the local Intranet
User Interface	The application shall provide a graphical visualization of the classified call records
	The application shall list all classified call records for detailed inspection
Usability	The application shall provide a consistent user interface across all the major desktop web browsers (i.e. Mozilla Firefox, Microsoft Internet Explorer, Google Chrome, and Opera).

3.5.2 System Design

System Model

The web application design consisted of three layers implemented in a web-based client-server model. The client-side was implemented in a web browser while the server-side was implemented in a web-server accessible across the organization's local area network. The top layer was the User Interface which provided the visualization. The middle layer was the Business Logic for call record classification. The bottom layer was the repository of Call Detail Record (CDR) files. This was the data layer. The application did not contain a relational database. Figure 3-18 shows the web application's 3-layer design.

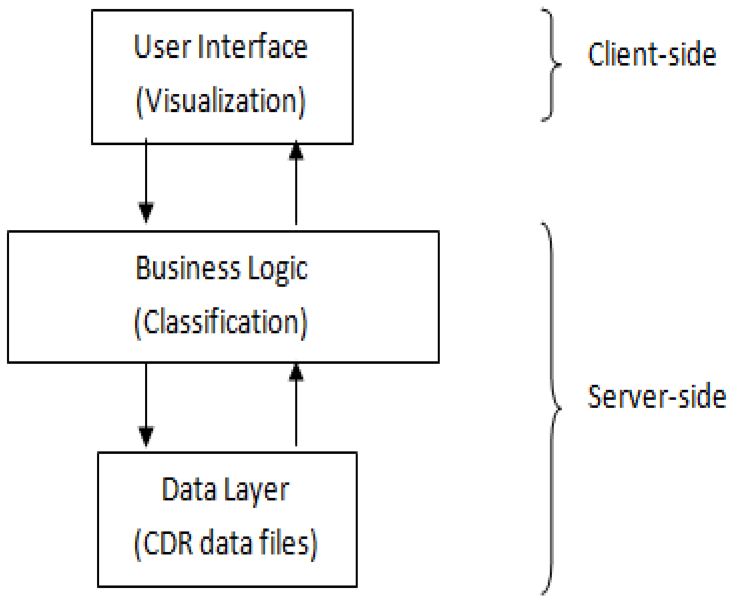


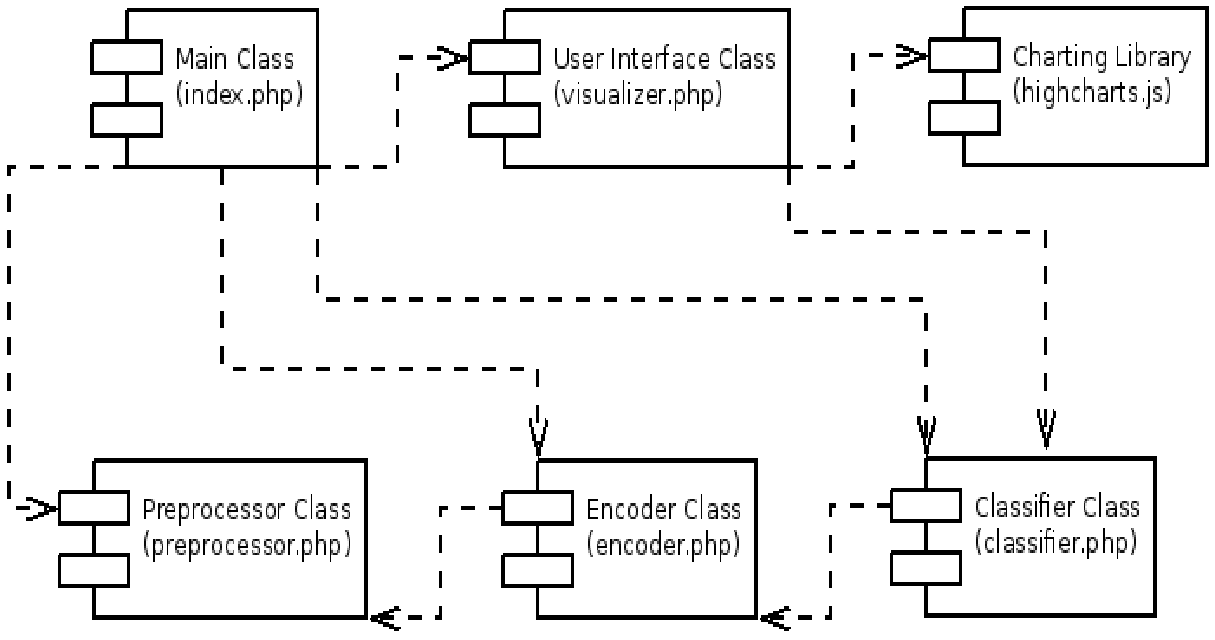
Figure 3-18:3-layer model of the web application

The User Interface layer was implemented in the client side while the Business Logic and Data Layer resided in the server side.

The communication between the top and middle layer used an HTTP web page request and response. The communication between the middle and bottom layer involved CDR file reads to extract the call records for classification.

System Modules

Six modules were identified for the web application. Figure 3-19 is the component diagram for the web application. It shows the six application modules and their dependencies.



Main Class (index.php)
User Interface Class (visualizer.php)
Charting Library (highcharts.js)
Preprocessor Class (preprocessor.php)
Encoder Class (encoder.php)
Classifier Class (classifier.php)

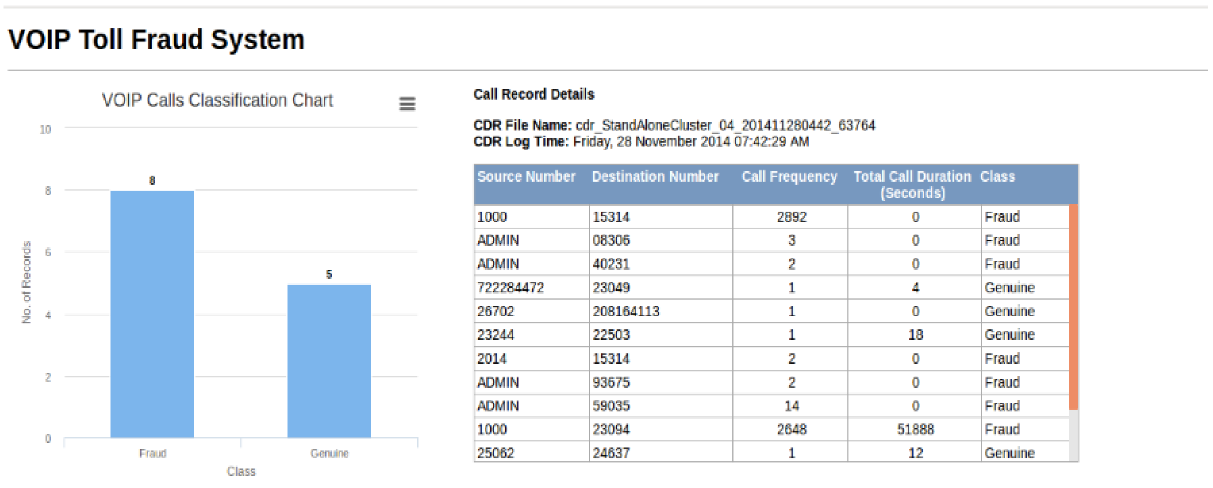
Figure 3-19: Component diagram for the web application

1. Main Class – This is the main module of the application. It calls the functions defined in the other modules when the user requests the web page.
2. Preprocessor Class – This module defines functions for preprocessing the CDR data file.
3. Encoder Class – This module defines functions for encoding the preprocessed CDR data file into a format that can be classified by the MATLAB neural network toolbox
4. Classifier Class – This module uses the trained MATLAB neural network object to classify the call records in the encoded file.
5. User Interface Class – This module prepares the classification output for visualization.
6. Charting Library – This module provides charting functionality.

3.5.3 Coding

The web application was implemented in PHP, JavaScript, HTML and CSS languages. HTML and CSS defined the User Interface layer. The Business Logic was coded in PHP hosted on an Apache Web Server. This included functionality for reading the CDR data files, preprocessing them, encoding, classifying and preparing the output for visualization. A third party JavaScript library known as Highcharts was used to construct a column chart for comparing the number of fraud and genuine records. An HTML table listed all the classified call records.

Figure 3-20 below shows the web application user interface. The visualized results of classified call records from a CDR file are shown on the web page. On the left hand side, a chart compares the number of calls classified as Fraud and Genuine. On the right-hand side, there is a table with each classified call record from the CDR file. Whenever the web page is refreshed, the application fetches the latest CDR file, classifies its records using the trained Matlab neural network and presents these visualized results to the user. The user is then able to quickly inspect any fraud records that have been flagged by the neural network.



Call Record Details

CDR File Name: cdr_StandAloneCluster_04_201411280442_63764
CDR Log Time: Friday, 28 November 2014 07:42:29 AM

Source Number	Destination Number	Call Frequency	Total Call Duration (Seconds)	Class
1000	15314	2892	0	Fraud
ADMIN	08306	3	0	Fraud
ADMIN	40231	2	0	Fraud
722284472	23049	1	4	Genuine
26702	208164113	1	0	Genuine
23244	22503	1	18	Genuine
2014	15314	2	0	Fraud
ADMIN	93675	2	0	Fraud
ADMIN	59035	14	0	Fraud
1000	23094	2648	51888	Fraud
25062	24637	1	12	Genuine

Figure 3-20: Web application's user interface

3.5.4 Testing

The software quality attributes and software quality metrics to be used as performance indicators during testing of the web application were identified at this stage. The attributes and metrics are presented in table 3-5 below.

Table 3-5: Web application functional requirements

Software Quality Attribute	Software Quality Metric
Completeness	% of specified functional requirements met by the application
Efficiency	Average web page load time (classification + visualization time)
Usability	The application provides a consistent user interface across all web browsers
Reliability	No. of system failures logged by the web server

The application is validated in a few different ways. The fraud cases that detected can be sent to the to an internal fraud investigation team, which can then make a final determination as to whether or not a particular case represents fraudulent activity. The fraud results of these investigation then allows for regular updates of the detection algorithms and since the fraud detection system is modular, it is easy to adjust a module and redeploy independently without compromising the integrity of the application. Another method of validation comes from the fact that the impact of fraud often shows up on the bills from VoIP service providers. Inconsistent billing reports from service providers will most likely point to an occurrence of VoIP fraud.

3.5.5 Deployment

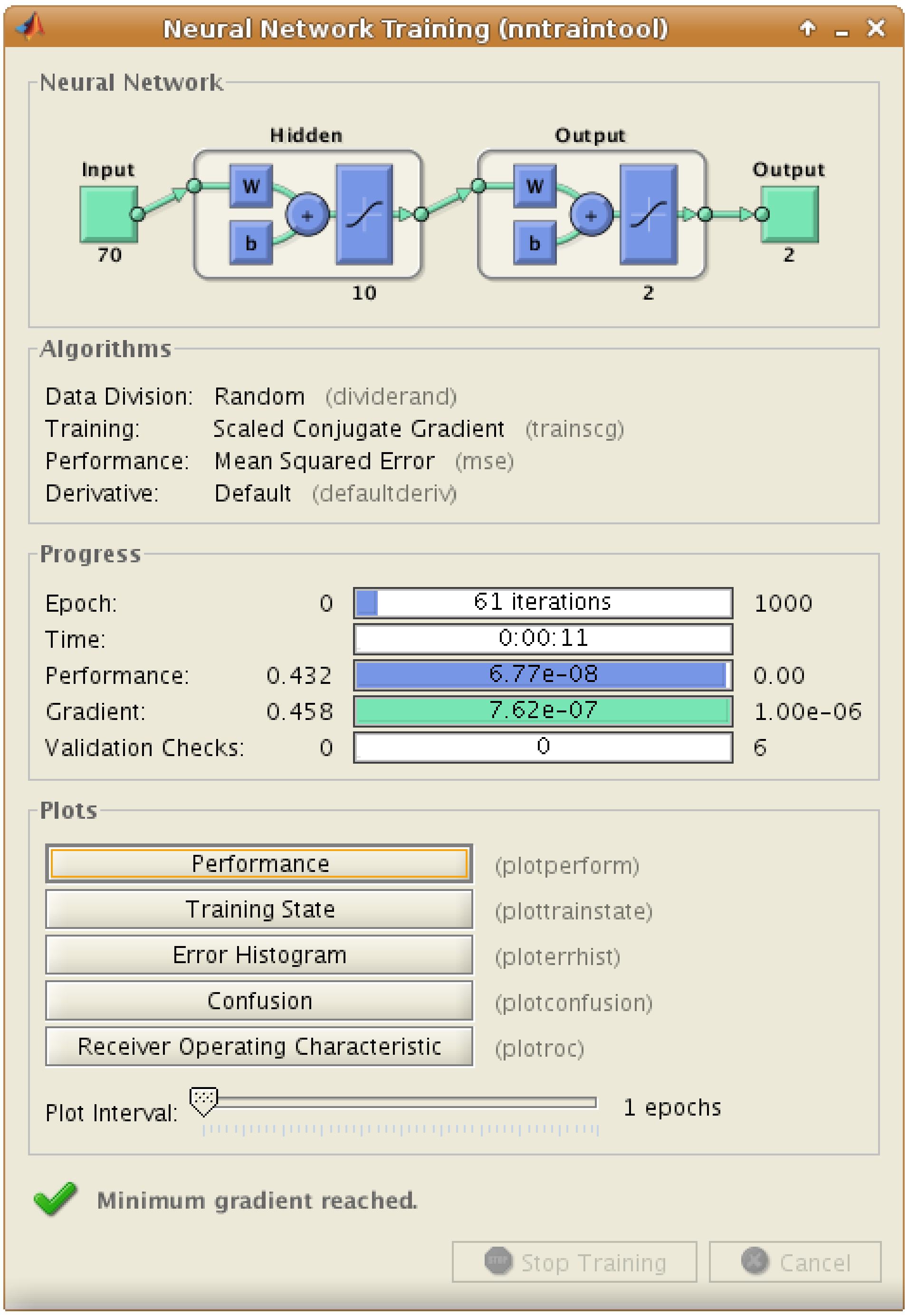
The web application will be deployed at an organization that is a seller or consumer of Voice over IP. The application will be deployed on the organization's Intranet or projected on a screen in the Network Operations Center (NOC) from where the administrators can monitor the VoIP network for any incidents of fraudulent behavior. The goal is to display all the call activity and not just the part that looks fraudulent thereby allowing the user to see the potentially fraudulent calls in the context of the normal calling pattern.

CHAPTER 4: RESULTS AND DISCUSIONS

This chapter describes in details the steps used to train and test the Network, experimentation process and the results obtained from the experiments.

4.1 Network Training Results

The neural network training ran for 61 epochs (iterations on the training data set) taking a period of 11 seconds as shown in Figure 4-1.



Neural Network Training (nntraintool)
Neural Network
Hidden
Output
Input
Output
W
b
W
b
70
10
2
2

Algorithms

```
Data Division:  Random  (dividerand)
Training:       Scaled Conjugate Gradient  (trainscg)
Performance:    Mean Squared Error  (mse)
Derivative:     Default  (defaultderiv)
```

Progress

```
Epoch:                  0   61 iterations   1000
Time:                       0:00:11
Performance:        0.432   6.77e-08        0.00
Gradient:           0.458   7.62e-07        1.00e-06
Validation Checks:      0   0               6
```

Plots
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Confusion (plotconfusion)
Receiver Operating Characteristic (plotroc)
Plot Interval:
1 epochs
Stop Training
Cancel

Minimum gradient reached.

Figure 4-1: Network training results

The performance measure used was the Mean Squared Error (MSE). MSE is the average squared difference between the class labels assigned by the network (outputs) and the actual class labels (targets). Lower MSE values are better and an MSE value of zero means no error.

Percent Error was also used as a performance measure alongside the MSE. It indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.

Table 4-1 (shown in Figure 3.10) shows the MSE and Percent Error results for the training, validation and testing data sets. It shows that the lowest MSE value was recorded on the validation data set while the highest MSE value was recorded on the testing data set. This is because the testing data set samples are not used during the training of the network.

All the data sets had very small MSE values. This indicates that the network training produced a network capable of high accuracy classification ability. In additions, the three data sets had a Percent Error of 0 meaning that there were no misclassifications in any of the data sets.

Table 4-1: MSE and Percent Error

	No. of Samples	**MSE**	**Percent Error**
Training	10,500	6.76675e-8	0
Validation	2,250	6.21602e-8	0
Testing	2,250	8.42228e-8	0

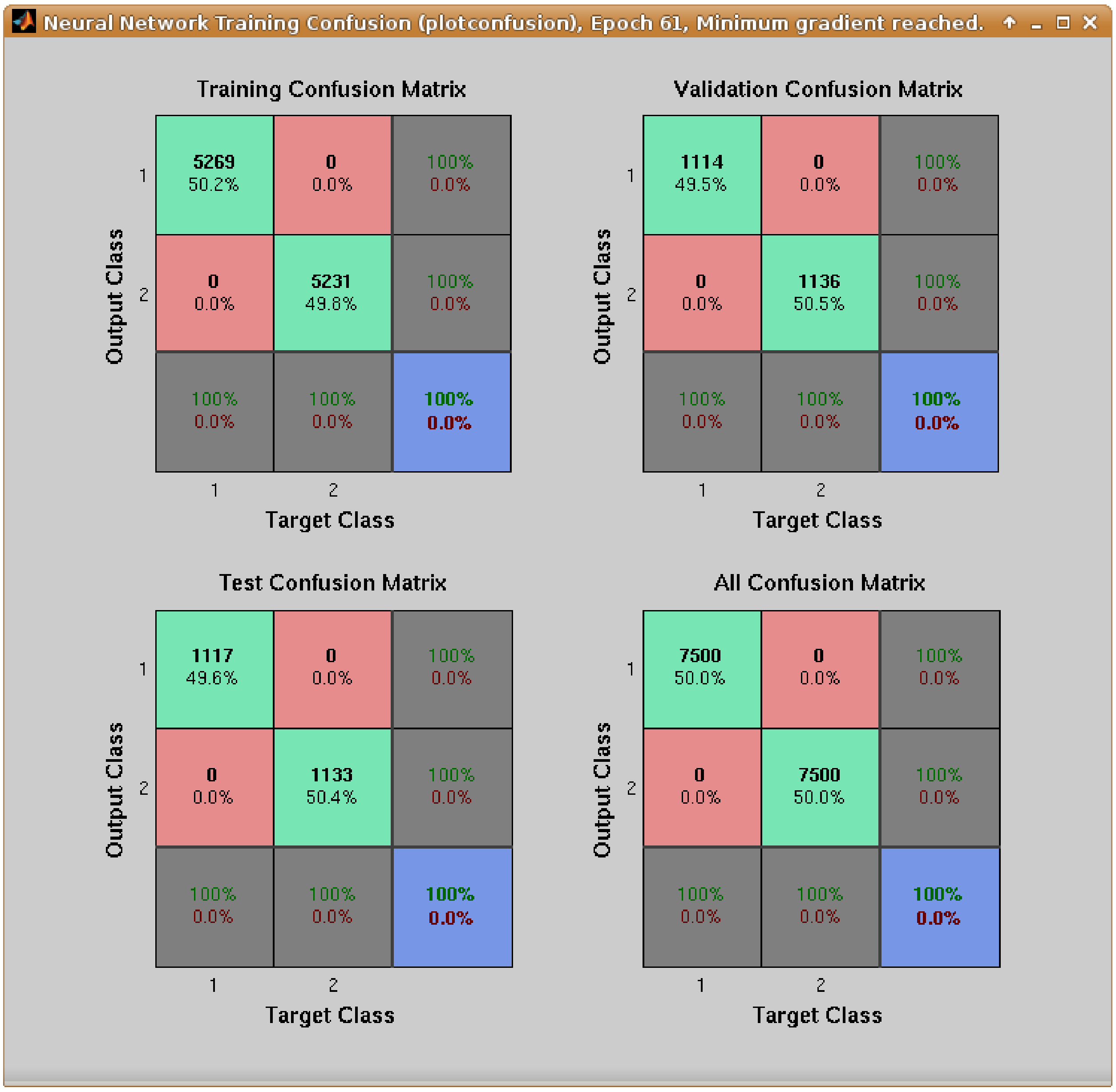
4.2 Confusion Matrices

The confusion matrices shown in Figure 4-2 below further elaborate these results. In the diagrams, 1 represents Fraud cases and 2 represents genuine cases. The Output Class is what is predicted by the network while the Target Class is the correct classification.

Crucially, the confusion matrices show that for the training, validation and testing data sets, all the samples were classified correctly. This is indicated by the value of 100% for correct classifications and 0% for incorrect classifications. The network outputs are very accurate, as seen by the maximum number of correct responses in the green squares and the minimum number of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

The neural network achieved the high classification accuracy because it was able to learn quite well the patterns of genuine calls that were used to label the sample data set. It was then able to extend this pattern to unseen instances of call records.

These results indicate that the features selected for the training were very relevant. The VoIP calls are well defined by the source number, destination number, call frequency, call duration and the time and day of the calls. These features are adequate for classifying the nature of calls as either being fraudulent or genuine.


Neural Network Training Confusion (plotconfusion), Epoch 61, Minimum gradient reached.

Training Confusion Matrix

Output Class \ Target Class	1	2	
1	5269 50.2%	0 0.0%	100% 0.0%
2	0 0.0%	5231 49.8%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%

Validation Confusion Matrix

Output Class \ Target Class	1	2	
1	1114 49.5%	0 0.0%	100% 0.0%
2	0 0.0%	1136 50.5%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%

Test Confusion Matrix

Output Class \ Target Class	1	2	
1	1117 49.6%	0 0.0%	100% 0.0%
2	0 0.0%	1133 50.4%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%

All Confusion Matrix

Output Class \ Target Class	1	2	
1	7500 50.0%	0 0.0%	100% 0.0%
2	0 0.0%	7500 50.0%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%

Figure 4-2: Confusion matrices

Figure 4-3 below shows how the MSE reduces to a minimum value of 6.216e-8 at epoch 61 of training on the validation data set. The validation data set achieved the minimum MSE at epoch 61. This is what informed the training process to stop at that point since the latter epochs had higher MSE values.

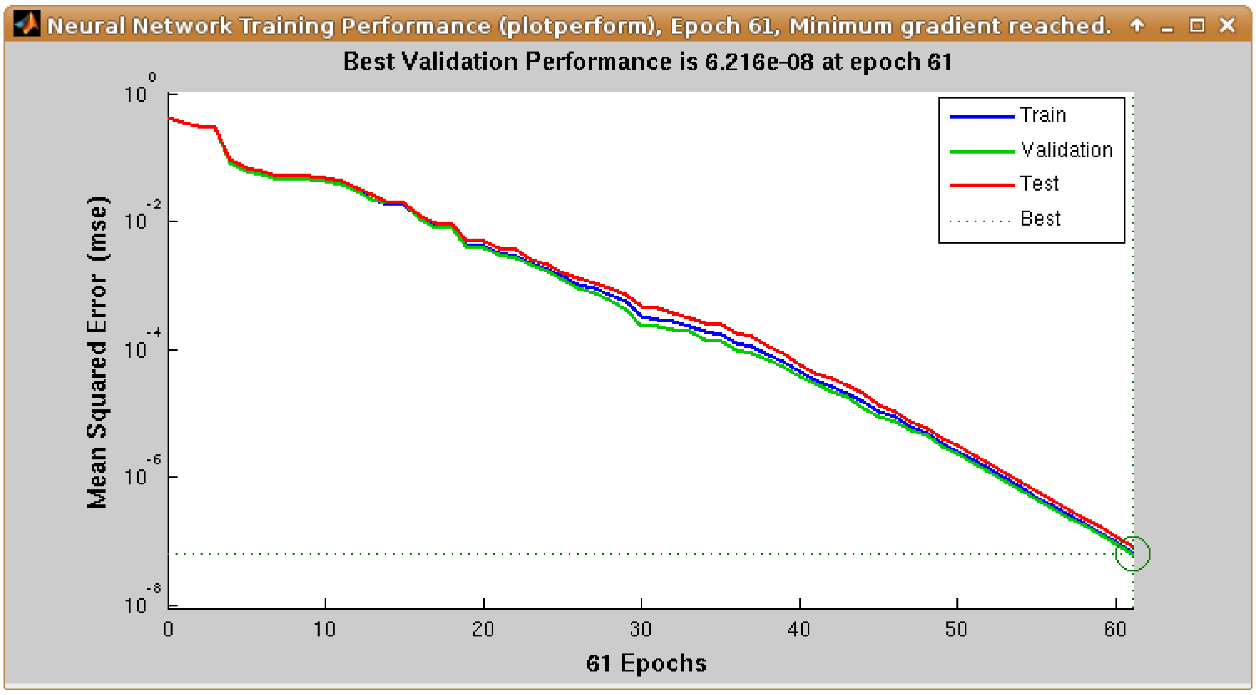


Figure 4-3: Neural network training performance on the validation data set

4.3 Receiver Operating Characteristic (ROC) Curves

The blue colored lines in each of the axes of Figure 4-4 represent the Receiver Operating Characteristic (ROC) curves. The ROC curves show the relationship between the False Positive Rate (1 - specificity) and True Positive Rate (sensitivity) for the three data sets. A perfect test would show points in the upper-left corner, with 100% sensitivity and 100% specificity.

The ROC curves here all show that the False Positive Rate remains at a minimum while the True Positive Rate rises to a maximum due to the 100% correct classification performance of the network. For this problem, the neural network performs very well.

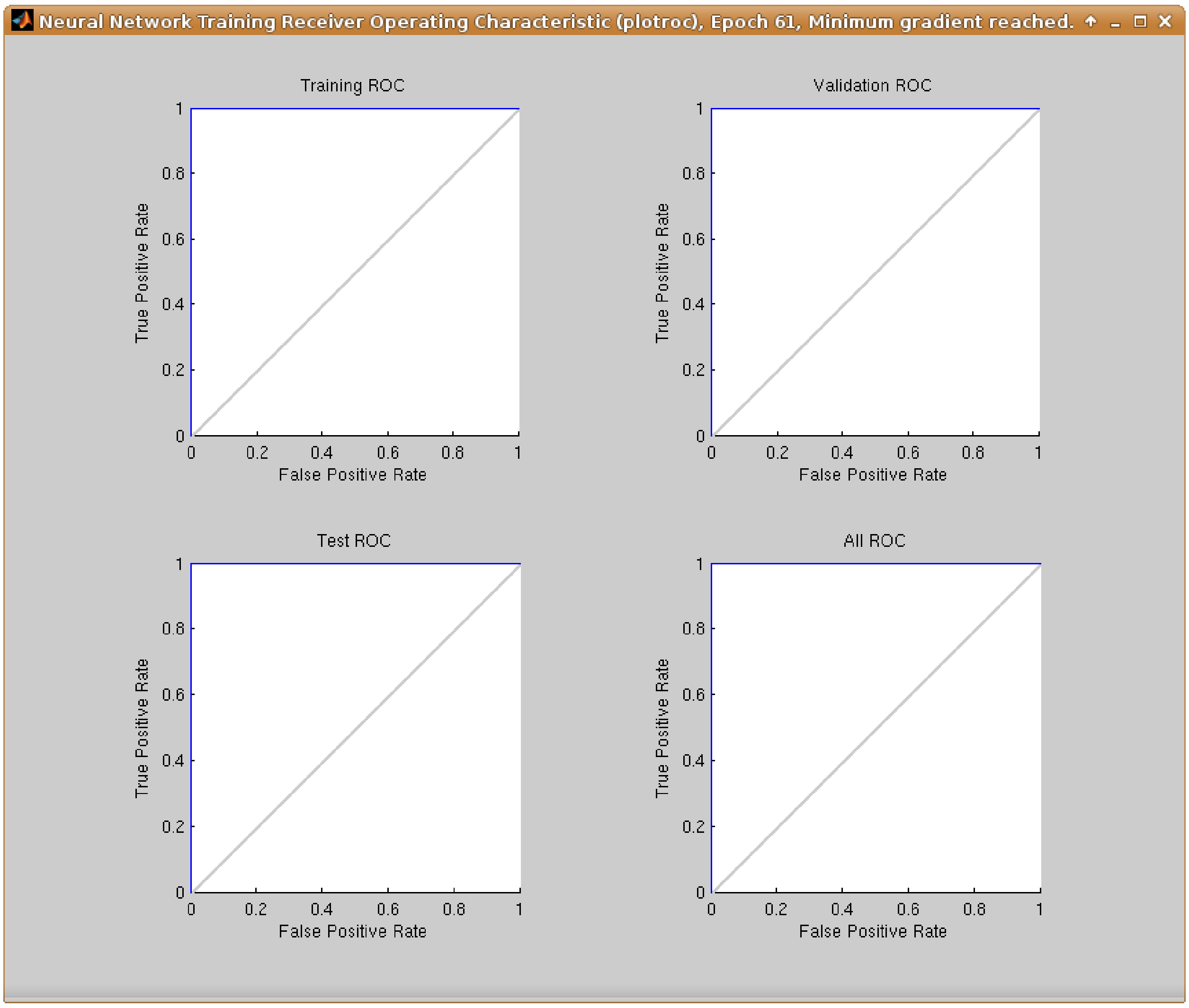


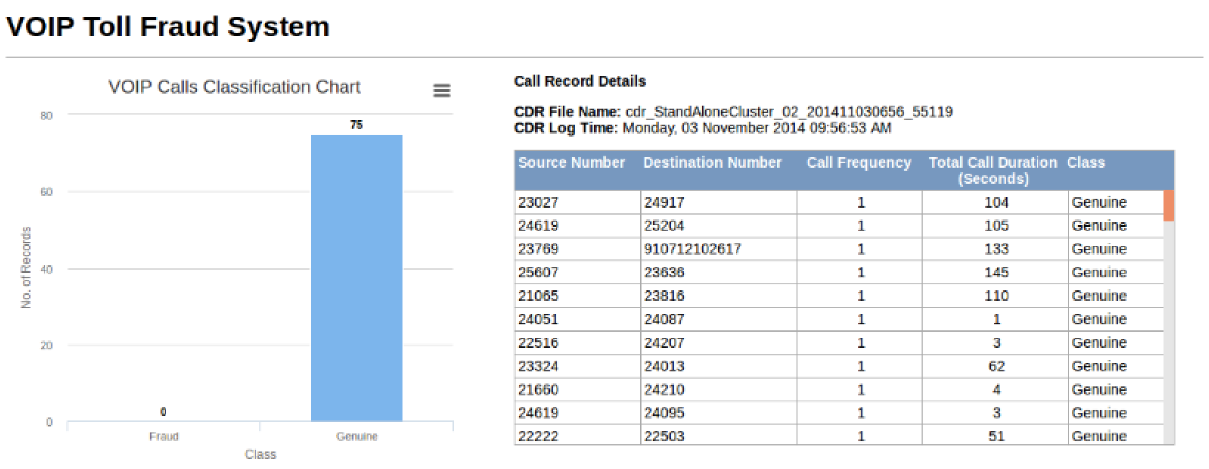
Figure 4-4: ROC plots

4.4 Analysis of the Results of the Fraud Detection System

VoIP fraud generally involves a third party making frequent long duration off-net calls at the expense of a business. VoIP callers would be classified as either fraudulent or genuine based on the following characteristics of the call:

1. Source Number- This attribute shows the number of the line that made the call. Legitimate callers should always bear the correct number within the company numbering plan.
2. Destination Number- This attribute specifies the number to which the call finally gets presented. This number should also be within the company's calling space for the outgoing call to be considered genuine.
3. Call Frequency-This is the number of calls made from a particular source to a particular destination within a specific CDR dump.
4. Total Call Duration- This is the total duration (in seconds) of calls made from a particular source to a particular destination within the specified CDR file.

The interfaces in figures 4-5, 4-6 and 4-7 show a sample classification and visualization report for selected CDR data. Figure 4-5 below shows an interface of the web application. Call records from the selected CDR file have all been classified genuine.



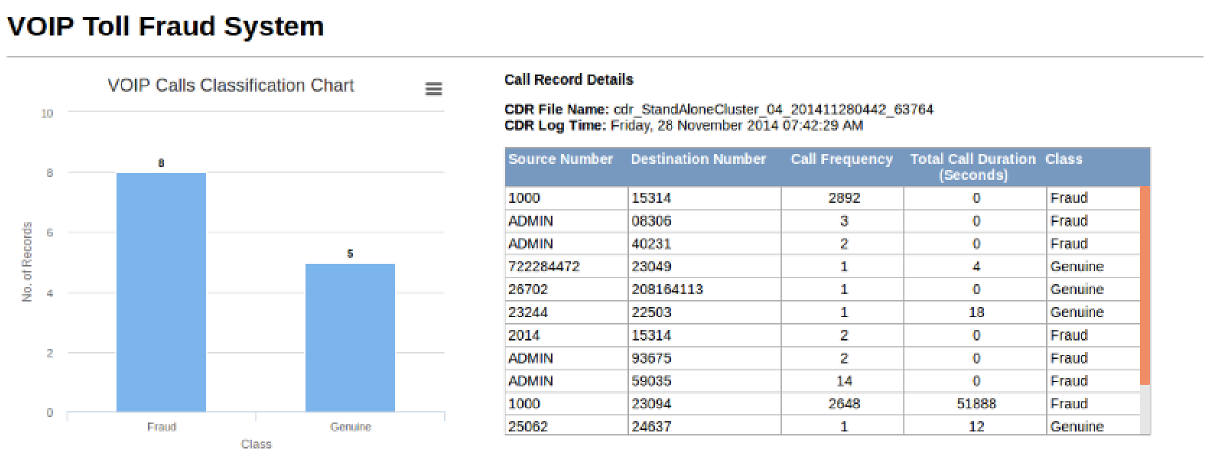
Call Record Details

CDR File Name: cdr_StandAloneCluster_02_201411030656_55119
CDR Log Time: Monday, 03 November 2014 09:56:53 AM

Source Number	Destination Number	Call Frequency	Total Call Duration (Seconds)	Class
23027	24917	1	104	Genuine
24619	25204	1	105	Genuine
23769	910712102617	1	133	Genuine
25607	23636	1	145	Genuine
21065	23816	1	110	Genuine
24051	24087	1	1	Genuine
22516	24207	1	3	Genuine
23324	24013	1	62	Genuine
21660	24210	1	4	Genuine
24619	24095	1	3	Genuine
22222	22503	1	51	Genuine

Figure 4-5: Web Application's Interface.

A number of callers have been flagged as being fraudulent callers in figure 4-6 below. The originating number of the calls does not conform to the expected originating numbers for calls. Additionally, a high frequency of calls is made within the duration of the CDR log file. These characteristics depict fraudulent behavior as earlier defined.



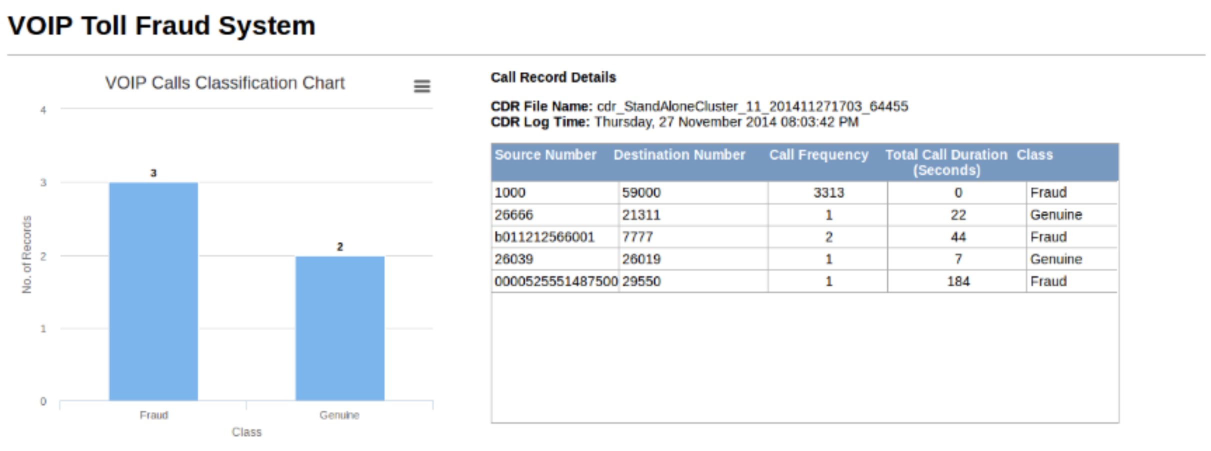
Call Record Details

CDR File Name: cdr_StandAloneCluster_04_201411280442_63764
CDR Log Time: Friday, 28 November 2014 07:42:29 AM

Source Number	Destination Number	Call Frequency	Total Call Duration (Seconds)	Class
1000	15314	2892	0	Fraud
ADMIN	08306	3	0	Fraud
ADMIN	40231	2	0	Fraud
722284472	23049	1	4	Genuine
26702	208164113	1	0	Genuine
23244	22503	1	18	Genuine
2014	15314	2	0	Fraud
ADMIN	93675	2	0	Fraud
ADMIN	59035	14	0	Fraud
1000	23094	2648	51888	Fraud
25062	24637	1	12	Genuine

Figure 4-6: Web Application’s Interface.

A number of callers have been flagged as being fraudulent callers in figure 4-7 below. The first call is marked by the system as being fraudulent for two reasons-the source number(1000) is outside the defined range of originating caller numbers and the high frequency of calls is made within duration (3313 calls). Similarly the second and fifth calls have been flagged as being fraudulent because their source numbers are outside the expected caller numbers.



Call Record Details

CDR File Name: cdr_StandAloneCluster_11_201411271703_64455
CDR Log Time: Thursday, 27 November 2014 08:03:42 PM

Source Number	Destination Number	Call Frequency	Total Call Duration (Seconds)	Class
1000	59000	3313	0	Fraud
26666	21311	1	22	Genuine
b011212566001	7777	2	44	Fraud
26039	26019	1	7	Genuine
0000525551487500	29550	1	184	Fraud

Figure 4-7: Web Application’s Interface.

CHAPTER 5: CONCLUSION

This chapter concludes the research and summarizes the achievements of the objectives along with the key findings of the research. In addition, this chapter also discusses the limitations encountered during the course of the research and provides key recommendations. It finally presents possible future research areas.

5.1 Achievements

The main objective of the study was to develop a Voice over IP fraud detection system that classifies VoIP calls as either genuine or fraudulent using Artificial Neural Networks. Actual Call Detail Records (CDR) collected from an operational VoIP system in Kenya was used in the study. The data set contained time series of call records from both fraudulent and legitimate callers. Feature selection was performed on the data in order to eliminate redundant variables and select the attributes that would best describe fraudulent behavior. The sampled data was then labeled as being fraudulent or genuine based on the attributes of the call. The data set was partitioned as follows- 70% of the data set was set aside for training, 15% for validation and the remaining 15% was the testing set. The training, validation and testing data sets consisted of both genuine and fraudulent call records. The destination phone numbers fell into the following categories: on-net (Internal) and off-net (Mobile, National and International).The implementation of the Artificial Neural Network was based on the Matlab Neural Network toolbox. The neural network achieved the high classification accuracy; it was able to learn quite well the patterns of genuine calls that were used to label the sample data set. It was then able to extend this pattern to unseen instances of call records. These results indicated that the features selected for the training were very relevant. The VoIP calls were defined by the source number, destination number, call frequency, call duration and the time and day of the calls. These features are adequate for classifying the nature of calls as either being fraudulent or genuine. The study established that Artificial Neural Network are a successful technology that can be applied in VoIP fraud detection since it was able to detect abrupt changes in established calling patterns which may be as a consequence of fraud.

A web application that reads and classifies call records in new CDR files was developed as part of the study. The application utilized the trained neural network and provided a visualization of the classified records. The application fetches the latest CDR file, classifies its records using the trained Matlab neural network and presents these visualized results.

The web application design consisted of three layers implemented in a web-based client-server model. The client-side was implemented in a web browser while the server-side was implemented in a web-server accessible across the organization's network. The top layer was the User Interface which provided the visualization. The middle layer was the Business Logic for call record classification. The bottom layer was the repository of Call Detail Record (CDR) files. This was the data layer. The User Interface layer was implemented in the client side while the Business Logic and Data Layer resided in the server side. The communication between the top and middle layer used an HTTP web page request and response. The communication between the middle and bottom layer involved CDR file reads to extract the call records for classification. The implementation of the fraud detection tool based on artificial intelligence will be a big step towards detection and mitigation of VoIP fraud.

5.2 Limitations

The following limitations were noted in the course of the research:

The ability of Artificial Neural Networks to identify indications of fraud is completely dependent on the accurate training; this demanded very high computer resources and takes a long time especially for cases with huge number of records. The training routine requires a very large amount of resource to ensure that the results are statistically accurate and are completed on time.

Since legitimate calls occur more often than fraudulent calls, there was a challenge in obtaining sufficient data for illegitimate calls. Fraudulent call data is relatively rare and the data collection involving human labor is time consuming.

5.3 Recommendations

VoIP fraud is, and will remain, a lucrative business. As VoIP continues to grow in popularity, schemes for beating the system will continue to become more complex and powerful. VoIP providers and enterprises must work together to ensure their networks are secure from any attacks by fraudsters. The risks of VoIP fraud can be lowered with secure VoIP networks and analyzing VoIP traffic for any signs of fraud. Below are some of the control measures that can be put in place.

For companies using VoIP services, administrators need to familiarize themselves with the company's calling patterns and monitor them regularly by reviewing call logs(even on a daily basis),this is an important step in detecting toll fraud. For companies, whose business is primarily domestic, a huge volume of international calls should raise an alarm. Companies that do make a lot of international calls should be aware of the countries where toll fraud most often occurs. Administrators should be on the lookout for any suspicious call activity after hours, including weekends and public holidays.

Companies need to implement calling restrictions (especially for international calls) or allow secured access. If your business makes a lot of international phone calls, consider adding an extra layer of security, such as an authorization code that must be input before placing an international call. These authorization codes should not be predictable numbers and should also be changed regularly. Supervisor and maintenance passwords should also be changed when the administrator, an employee, or a contractor leaves the business.

Companies using VoIP services need to set up a firewall that help protect VoIP systems from fraud. A firewall, which inspects both voice and data packets as they pass through the network, can be used as a filter for fraudulent calling. The firewall must be configured with access rules that only ensure access is open to only authorized IP addresses. Since fraudsters scam VoIP systems for available open ports, companies should avoid leaving open ports for remote access to the VoIP system unless it is absolutely necessary.

Companies that use VoIP services may also make an arrangement with their providers to enable them get notifications from their provider should there be suspicious/unusual usage associated with their channels. Where possible, companies should consider using prepaid billing or postpaid with quotas that will allow you to terminate service in the event of a break in.

Companies should also develop alert systems for unusual calling patterns that monitors the VoIP traffic and notifies the administrator if traffic patterns seem unusual. The earlier the violation is detected the sooner that it can be closed meaning that the loss will be minimal. Having automated reports or more advanced unusual traffic pattern detection is a good addition.

5.4 Future Work

Fraud detection is only the first step in the mitigation of VoIP fraud, future work need to be focused on VoIP fraud prevention especially considering the amount of money lost by VoIP providers and enterprises using VoIP services when are hit by fraudsters.

We also recommend that future research in this area combines different data for data mining and even use more than one model each with different algorithm to mine data. Comparing the results obtained by each model might increase confidence in the results.

REFERENCES

Aggarwal,C.& Yu,P,.(2007). Outlier Detection for High Dimensional Data. In Proceedings of ACM

Barson, P., Field, S., Davey, N., McAskie, G. & Frank, R. (1996). The detection of fraud in mobile phone networks. *Neural Network World*, 6(4):477–484.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. Machine Learning , 79 (1-2), 151-175.

Blavette, V (2001). Application of intelligent techniques to telecommunication fraud detection. In European Institute for Research and Strategic Studies in Telecommunications, Public Project.

Bolton, R, J & Hand, J, D (2001). Unsupervised Profiling Methods for Fraud Detection, Technical Report, Imperial College.

Bolton, R. & Hand, D. (2002). Statistical Fraud Detection: A Review (With Discussion). Statistical Science 17(3): 235-255

Cortes, C., Pregibon, D. (2007). Giga-mining. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining;New York, NY:AAAI Press, 2007

Encyclopedia of Mathematics. (2013). Machine Learning. Retrieved June 18, 2013, from http://www.encyclopediaofmath.org/index.php/Machine_learning.

Flexer, A. (1999). On the use of self-organizing maps for clustering and visualization. In *Principles of Data Mining and Knowledge Discovery*,80–88.

Flanagan, J. A. (2001). Self-organization in the one-dimensional SOM with a decreasing neighborhood. *Neural Networks*, 14(10):1405–1417.

Folasade I. O. (2011). Computational Intelligence in Data Mining and Prospects in Telecommunication Industry. *Journal of Emerging Trends in Engineering and Applied Sciences. 2(2):*601–605.

Frank R.J.,Hunt,S.P& Davey,N (1999). Applications of Neural Networks to Telecommunications Systems, *Proc. of EUFIT '99*, 255,[ISBN 3-89653-808-X],Department of Computer Science, University of Hertfordshire, United Kingdom.

Frisone, F., Morasso, P., & Perico, L. (1998). Self-organization in cortical maps & EM learning. *Journal of Advanced Computational Intelligence*, 2:178–184.

Gary. M. W,(2005). Data Mining in Telecommunications. Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers. Kluwer Academic Publishers:1189-1201.

Ghaffari, A., Abdollahi, H., Khoshayand, M. R., Bozchalooi, I. S., Dadgar, A., & Refiee-Tehrani, M. (2006). Performance Comparison of Neural Network Training Algorithms in Modeling of Bimodal Drug Delivery. International Journal of Pharmaceutics , 327 (1-2), 126-138.

Giles, L. C., Lawrence, S., & Tsoi, C. A. (2001). Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference. Machine Learning , 44 (1-2), 161-183.

Gomes, G. S., Ludermir, T. B., & Lima, L. M. (2011). Comparison of New Activation Functions in Neural Network for Forecasting Financial Time Series. Neural Computing and Applications , 20 (3), 417-439.

Han, S., & Cho, S. (2006). Evolutionary Neural Networks for Anomaly Detection Based on the Behavior of a Program. IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics , 36 (3), 559-570.

Hoath, P. (1998). Telecoms fraud, the gory details. "Computer Fraud & Security"*20* (1), 10–14.

Keogh, E., & Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks - A Survey and Empirical Demonstration. Data Mining and Knowledge Discovery , 7 (4), 349–371.

Kim,S., Cho,N.W. & Lee Y. J. (2013). Application of density-based outlier detection to database activity monitoring, Information Systems Frontiers,15:55–65.Kohonen, T. (2001) Self-Organizing Maps, Berlin, Germany: Springer-Verlag.

Lin, S. and Si, J. (1998). Weight-value convergence of the SOM algorithm for discrete input. Neural Computation, 10(4):807–14.

Moreau, Y., Herman,V, & Vandewalle,J(1997). Detection of mobile phone fraud using supervised neural networks: A first prototype. In International Conference on Artificial Neural Networks Proceedings (ICANN'97):1065–1070.

Oja,M, Kaski,S & Kohonen,T.(2001). Bibliography of Self-Organizing Map (SOM) papers: 1 Addendum, Helsinki University of Technology, Neural Networks Research Centre.
Olszewski, D. (2012) A probabilistic approach to fraud detection in telecommunications*"* Knowledge-Based Systems, 26:246–258

Ortiz-Rodriguez, J. M., Martinez-Blanco, M. R., Varamontes, J. M., & Vega-Carrillo, H. R. (2013). Robust Design of Artificial Neural Networks Methodology in Neutron Spectrometry. (K. Prof. Suzuki, Ed.) Artificial Neural Networks - Architectures and Applications.

Pal, J. K (2011). Usefulness and applications of data mining in extracting information from different perspective, *Annals of Library and Information Studies*,58: 7-16.

Rosset,S., Murad,U, E.,Neumann,E,Y. & Pinkas,G (1999). Discovery of fraud rules for telecommunications—challenges and solutions, in Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,409–413, ACM, San Diego, Calif, USA.

Ruiz-Agundez,I, Penya Y. K., & P. Garcia B,(2010). Fraud detection for voice over IP services on next-generation networks,in Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices.

Russell, S., & Norvig, P. (2009). Artificial Intelligence - A Modern Approach (3rd ed.). New Jersey: Prentice Hall.

Samarati, P, Tunstall, M, Posegga, & Sauveron, D.(2010). Lecture Notes in Computer Science, 6033:199–212.

SI Cortes, C. & Pregibon, D (2001). Signature-based methods for data streams. Data Mining and Knowledge Discovery, 5(3):167-182

The Kenya Cyber Security Report 2014.Retrived July 20, 2015 from http://www.serianu.com/downloads/KenyaCyberSecurityReport2014.pdf

The State of Phone Fraud(2012). Retrieved July 21, 2015, from https://www.pindrop.com/phone-fraud-report/

The Telcom Fraud Survey 2015.Retrieved July 19,2015 from http://cfca.org/fraudlosssurvey/2015.pdf

Tobin, P.K.J. & Bidoli, M. (2006). Factors affecting the adoption of Voice over Intemet Protocol (VoIP) and other converged IP services in South Africa. South African Journal of Business Management, 37(1), 31 - 40.

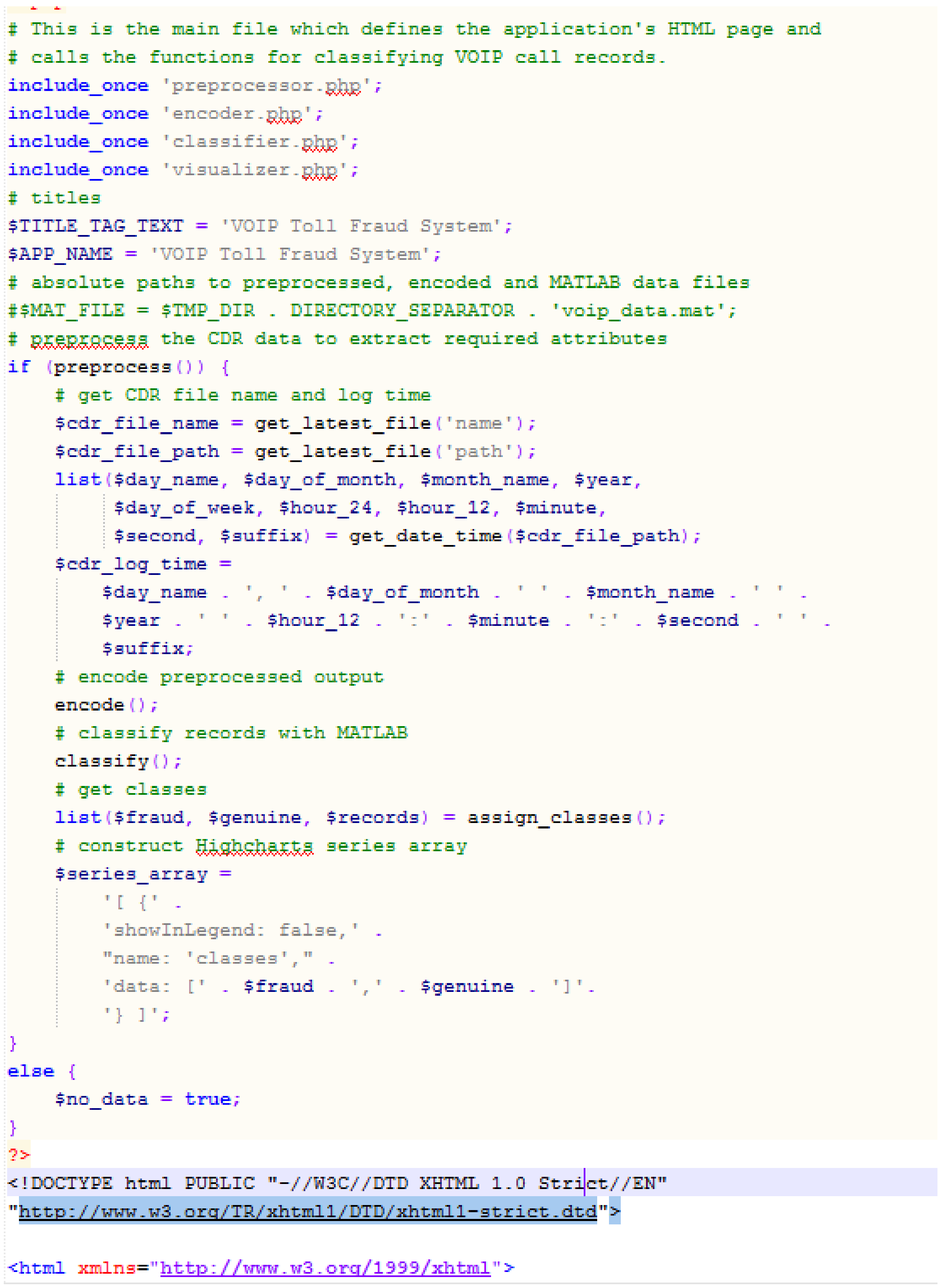
Tulankar,K., Kshirsagar,M.,& Wajgi,R.(2012).Clustering Telecom Customers using Emergent Self Organizing Maps for Business Profitability. *International Journal of Computer Science and Technology,* 3(1):256-259.

VoIP fraud Whitepaper (2015). Retrieved July 19, 2015, from
http://transnexus.com/resources/telecom-industry-topics/fraud-detection/download-voip-fraud-white-paper/

Weiss, G.M. (2009).Data mining in the Telecommunications Industry, IGI Global – Section: Service.

Yamanishiand J. T, (2001). Discovering Outlier Filtering Rules from Unlabeled Data - Combining a Supervised Learner with an Unsupervised Learner. In Proceedings of ACM (KDD 2001), San Francisco, California, USA.

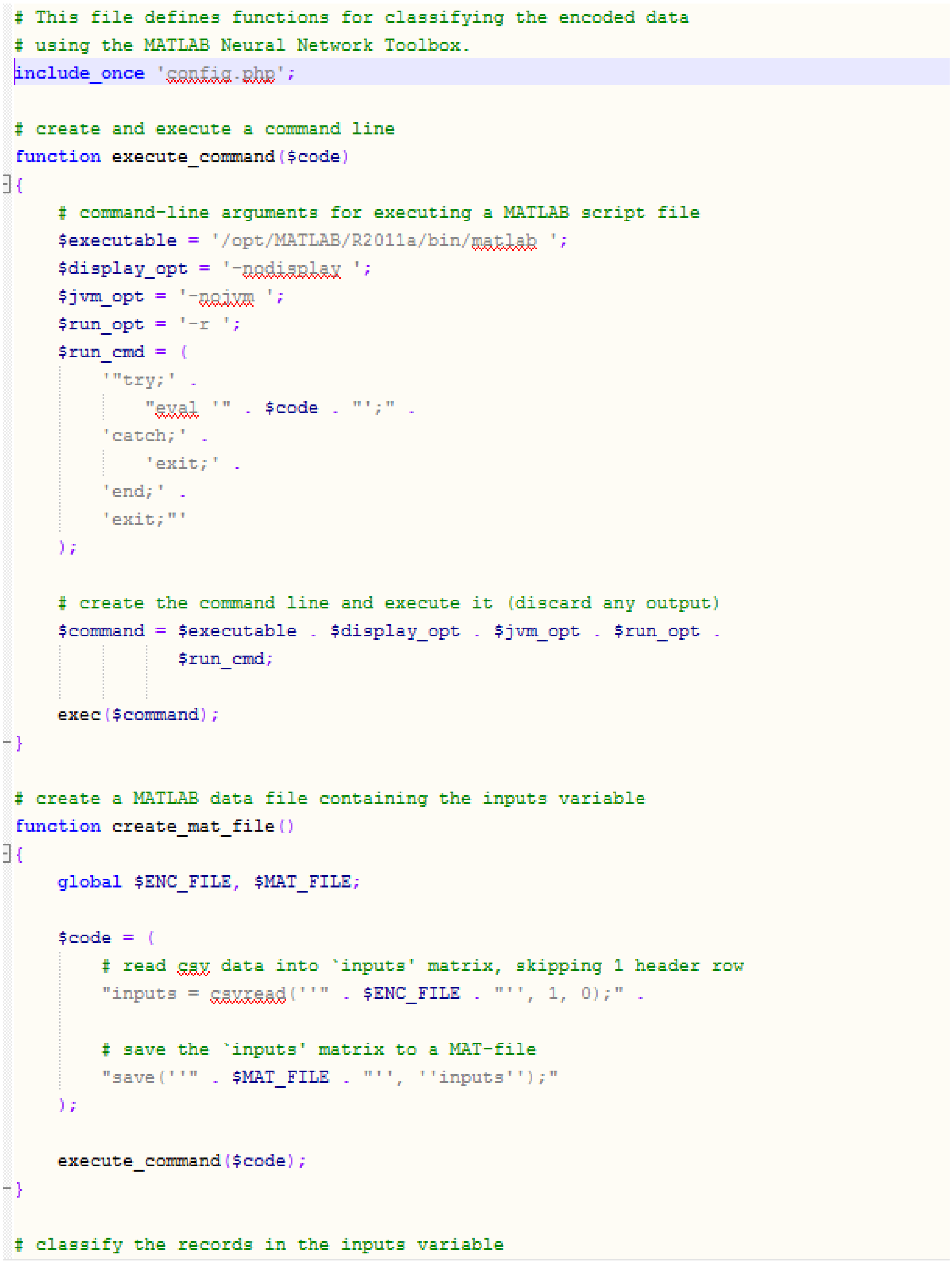
Appendix A: Index.php



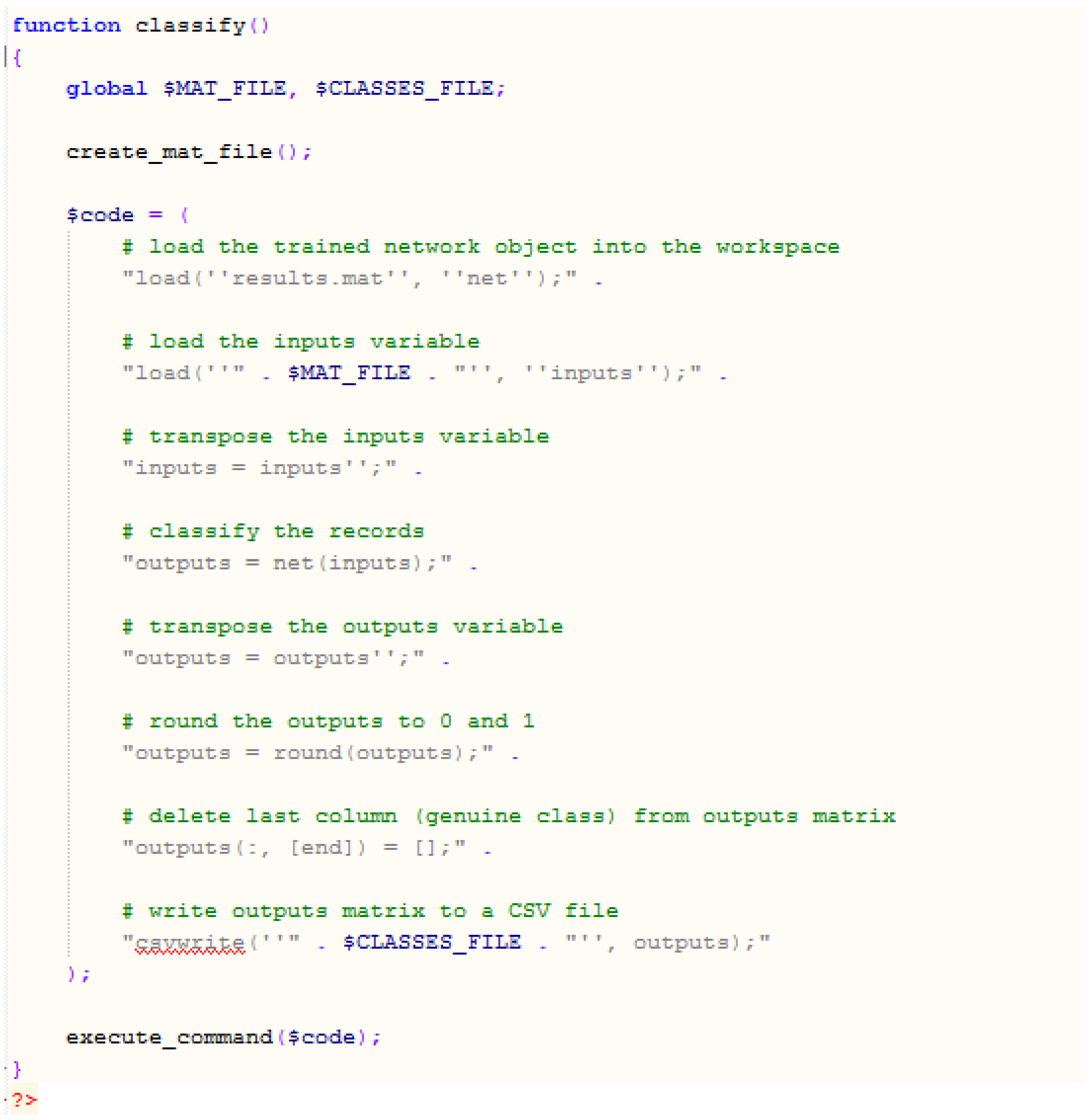
```
# This is the main file which defines the application's HTML page and
# calls the functions for classifying VOIP call records.
include_once 'preprocessor.php';
include_once 'encoder.php';
include_once 'classifier.php';
include_once 'visualizer.php';
# titles
$TITLE_TAG_TEXT = 'VOIP Toll Fraud System';
$APP_NAME = 'VOIP Toll Fraud System';
# absolute paths to preprocessed, encoded and MATLAB data files
#$MAT_FILE = $TMP_DIR . DIRECTORY_SEPARATOR . 'voip_data.mat';
# preprocess the CDR data to extract required attributes
if (preprocess()) {
    # get CDR file name and log time
    $cdr_file_name = get_latest_file('name');
    $cdr_file_path = get_latest_file('path');
    list($day_name, $day_of_month, $month_name, $year,
         $day_of_week, $hour_24, $hour_12, $minute,
         $second, $suffix) = get_date_time($cdr_file_path);
    $cdr_log_time =
        $day_name . ', ' . $day_of_month . ' ' . $month_name . ' ' .
        $year . ' ' . $hour_12 . ':' . $minute . ':' . $second . ' ' .
        $suffix;
    # encode preprocessed output
    encode();
    # classify records with MATLAB
    classify();
    # get classes
    list($fraud, $genuine, $records) = assign_classes();
    # construct Highcharts series array
    $series_array =
        '[ {' .
        'showInLegend: false,' .
        "name: 'classes'," .
        'data: [' . $fraud . ',' . $genuine . ']'.
        '} ]';
}
else {
    $no_data = true;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <title><?php echo $TITLE_TAG_TEXT; ?></title>
    <script src="js/jquery-1.11.3.min.js"></script>
    <script src="js/highcharts.js"></script>
    <script src="js/exporting.js"></script>
    <script src="js/offline-exporting.js"></script>
    <script src="js/main.js"></script>
</head>
<body>
    <!-- title div -->
    <div id="title">
        <h1><?php echo $APP_NAME; ?></h1>
    </div>
    <?php if (isset($no_data)) { ?>
        <!-- No Data div -->
        <div class="no_data">
            <h4>No Data Available!</h4>
            <p>The latest CDR file does not contain any call records.</p>
        </div>
    <?php } else { ?>
        <!-- chart div -->
        <div id="chart">
            <div id="container" class="highchart"></div>
            <script type="text/javascript">
                /* set highchart global options */
                set_highchart_options();
                /* plot options */
                var id = 'container';
                var type = 'column';
                var title = 'VOIP Calls Classification Chart';
                var subtitle = '';
                var xlabel = 'Class';
                var xlabels_array = ['Fraud', 'Genuine'];
                var ylabel = 'No. of Records';
                var yformat = '{value:,.0f}';
                var value_suffix = '';
                var series_color = '#FF8000';
                var series_array = <?php echo $series_array; ?>;
                /* plot highchart */
                plot_highchart(
                    id, type, title, subtitle, xlabel, xlabels_array,
                    ylabel, yformat, value_suffix, series_color,
                series_array
            );
        </script>
    </div>
    <!-- records table div -->
    <div id="records">
        <p><b></>Call Record Details</b></p>
        <p>
            <b>CDR File Name:</b> <?php echo $cdr_file_name; ?> <br />
            <b>CDR Log Time:</b> <?php  echo $cdr_log_time; ?>
        </p>
        <table>
            <thead>
                <tr>
                    <th class="src">Source Number</th>
                    <th class="dest">Destination Number</th>
                    <th class="freq">Call Frequency</th>
                    <th class="dur">Total Call Duration<br />(Seconds)</th>
                    <th class="classification">Class</th>
                </tr>
            </thead>

            <tbody>
                <?php echo $records; ?>
            </tbody>

        </table>
    </div>

<?php } ?>

</body>

</html>
```

Appendix B:Classifier.php



```php
# This file defines functions for classifying the encoded data
# using the MATLAB Neural Network Toolbox.
include_once 'config.php';

# create and execute a command line
function execute_command($code)
{
    # command-line arguments for executing a MATLAB script file
    $executable = '/opt/MATLAB/R2011a/bin/matlab ';
    $display_opt = '-nodisplay ';
    $jvm_opt = '-nojvm ';
    $run_opt = '-r ';
    $run_cmd = (
        '"try;' .
            "eval '" . $code . "';" .
        'catch;' .
            'exit;' .
        'end;' .
        'exit;"'
    );

    # create the command line and execute it (discard any output)
    $command = $executable . $display_opt . $jvm_opt . $run_opt .
                $run_cmd;

    exec($command);
}

# create a MATLAB data file containing the inputs variable
function create_mat_file()
{
    global $ENC_FILE, $MAT_FILE;

    $code = (
        # read csv data into `inputs' matrix, skipping 1 header row
        "inputs = csvread(''" . $ENC_FILE . "'', 1, 0);" .

        # save the `inputs' matrix to a MAT-file
        "save(''" . $MAT_FILE . "'', ''inputs'');"
    );

    execute_command($code);
}

# classify the records in the inputs variable
```



```
function classify()
{
    global $MAT_FILE, $CLASSES_FILE;

    create_mat_file();

    $code = (
        # load the trained network object into the workspace
        "load(''results.mat'', ''net'');" .

        # load the inputs variable
        "load(''" . $MAT_FILE . "'', ''inputs'');" .

        # transpose the inputs variable
        "inputs = inputs'';" .

        # classify the records
        "outputs = net(inputs);" .

        # transpose the outputs variable
        "outputs = outputs'';" .

        # round the outputs to 0 and 1
        "outputs = round(outputs);" .

        # delete last column (genuine class) from outputs matrix
        "outputs(:, [end]) = [];" .

        # write outputs matrix to a CSV file
        "csvwrite(''" . $CLASSES_FILE . "'', outputs);"
    );

    execute_command($code);
}
?>
```