An Ensemble Self-Structuring Neural Network Approach to Solving Classification Problems with Virtual Concept Drift and its Application to Phishing Websites

By

Rami Mustafa A Mohammad

A Thesis submitted to the University of Huddersfield
In partial fulfilment of the requirements for
The degree of Doctor of Philosophy

School of Computing and Engineering

University of Huddersfield

Huddersfield, United Kingdom

February 2016

Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the Copyright) and s/he has given The University of Huddersfield the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

ii. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

iii. The ownership of any patents, designs, trademarks and any all other intellectual property rights except for the Copyright (the Intellectual Property Rights) and any reproductions of copyright works, for example graphs and tables (Reproductions), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

Acknowledgments

Completing a doctoral dissertation is undoubtedly a demanding undertaking.

First and foremost, I thank God for giving me the strength and ability to pursue this goal in my life.

Grateful thanks to my mother Lutfiah and my father Mustafa for all their prayers.

To my wife Maha, I extend my gratitude for her full support and for managing our family while I concentrated on my studies.

To my beloved daughter Bana and to my wonderful sons Laith and Basel, I extend my love and hope that each of them will one day complete their doctorate and thus contribute significantly to the progress of humankind.

To my immediate family members, I extend my greatest love and gratitude for their unwavering support and the sacrifices they made during my doctoral program.

To my main supervisor Professor Thomas Lee McCluskey, I owe a special word of appreciation for his support. My heartfelt appreciation and gratitude are extended to my local supervisor Dr. Fadi Thabtah.

To those people who stood in my way as an obstacle and tried to dissuade me from continuing with my PhD, irrespective of their motives, I thank you for indirectly giving me even more passion, strength and determination to put my doctoral studies first and give my thesis the utmost care and attention.

Publications

Most of the chapters in this thesis have been published or submitted for publication in refereed international journals or conferences. The published papers are given below.

Journals

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2013-A. Intelligent Rule based Phishing Websites Classification. *IET Information Security,* 8 (3), pp. 153-160.

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2013-B. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications (Springer),* 25 (2), pp. 443-458.

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2015-A. Tutorial and critical analysis of phishing websites methods. *Computer Science Review-(ELSEVIER)*, 17(1), pp.1-24.

- Mohammad, R. M., McCluskey, T.L. & Thabtah, F., 2016-A. An ensemble self-structuring neural network approach and its application to to phishing websites. *Neural Computing and Applications*.

Conferences

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2012. *An assessment of features related to phishing websites using an automated technique.* London, IEEE, pp. 492-497.

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2013-C. *Predicting Phishing Websites using Neural Network trained with Back-Propagation.* Las Vigas, World Congress in Computer Science, Computer Engineering, and Applied Computing, pp. 682-686.

- Mohammad, R. M., Thabtah, F. & McCluskey, L., 2016-B. *An Improved Self-Structuring Neural Network*. In Pacific Asia Knowledge Discovery and Data Mining Conference (PAKDD) 2016. Auckland, 2016 B. Springer (LNCS).

Abstract

Keywords: *Phishing, Data Mining, Neural Network, Classification, Ensemble, Concept Drift, Incremental Learning, Self-Structuring, Website Features*

Classification in data mining is one of the well-known tasks that aim to construct a classification model from a labelled input data set. Most classification models are devoted to a static environment where the complete training data set is presented to the classification algorithm. This data set is assumed to cover all information needed to learn the pertinent concepts (rules and patterns) related to how to classify unseen examples to predefined classes. However, in dynamic (non-stationary) domains, the set of features (input data attributes) may change over time. For instance, some features that are considered significant at time T_i might become useless or irrelevant at time T_{i+j}. This situation results in a phenomena called Virtual Concept Drift. Yet, the set of features that are dropped at time T_{i+j} might return to become significant again in the future. Such a situation results in the so-called Cyclical Concept Drift, which is a direct result of the frequently called catastrophic forgetting dilemma. Catastrophic forgetting happens when the learning of new knowledge completely removes the previously learned knowledge.

Phishing is a dynamic classification problem where a virtual concept drift might occur. Yet, the virtual concept drift that occurs in phishing might be guided by some malevolent intelligent agent rather than occurring naturally. One reason why phishers keep changing the features combination when creating phishing websites might be that they have the ability to interpret the anti-phishing tool and thus they pick a new set of features that can circumvent it. However, besides the generalisation capability, fault tolerance, and strong ability to learn, a Neural Network (NN) classification model is considered as a black box. Hence, if someone has the skills to hack into the NN based classification model, he might face difficulties to interpret and understand how the NN processes the input data in order to produce the final decision (assign class value).

In this thesis, we investigate the problem of virtual concept drift by proposing a framework that can keep pace with the continuous changes in the input features. The proposed framework has been applied to phishing websites classification problem and it shows competitive results with respect to various evaluation measures (Harmonic Mean (F1-score), precision, accuracy, etc.) when compared to several other data mining techniques. The framework creates an ensemble of classifiers (group of classifiers) and it offers a balance between stability (maintaining previously learned knowledge) and plasticity (learning knowledge from the newly offered training data set). Hence, the framework can also handle the cyclical concept drift. The classifiers that constitute the ensemble are created using an improved Self-Structuring Neural Networks algorithm (SSNN). Traditionally, NN modelling techniques rely on trial and error, which is a tedious and time-consuming process. The SSNN simplifies structuring NN classifiers with minimum intervention from the user. The framework evaluates the ensemble whenever a new data set chunk is collected. If the overall accuracy of the combined results from the ensemble drops significantly, a new classifier is created using the SSNN and added to the ensemble. Overall, the experimental results show that the proposed framework affords a balance between stability and plasticity and can effectively handle the virtual concept drift when applied to phishing websites classification problem. Most of the chapters of this thesis have been subject to publication.

List of Acronyms

2FA ..... Two-Factor-Authentication

ADWIN ..... Adaptive Sliding Window

AGA ..... Adaptive Genetic Algorithms

AI ..... Artificial Intelligence

AIWL ..... Automated Individual Whitelist

AOL ..... America Online

APWG ..... Anti-Phishing Working Group

BN ..... Bayesian networks

CA ..... Certification Authority

CANTINA ..... Carnegie Mellon Anti-phishing and Network Analysis Tool

ccTLD ..... Country Code Top Level Domains

DM ..... Data Mining

DMOZ ..... Directory Mozilla

DNS ..... Domain Name System

DT ..... Decision Tree

EMD ..... Earth Mover's Distance

ESSNN ..... Ensemble Self-Structuring Neural Network

FIFO ..... First In First Out

FLORA ..... Floating Rough Approximation

FN ..... False Negative

FFNN ..... Feed-Forward Neural Network

FP ..... False Positive

FRANN ..... Floating Rough Approximation in Neural Network

FTC ..... Federal Trade Commission

GA.....Genetic Algorithms

FTP..... File Transfer Protocol

HAT .....Hoeffding Adaptive Tree

Htaccess ..... Hypertext Access

HTTPS..... Hyper-Text Transfer Protocol with Secure Sockets Layer

IBPA..... Instantaneous Based Protection Approach

IG ..... Information Gain

IM.....Instant Messaging

IRC.....Internet Relay Chat

LR.....Logistic Regression

MDL..... Minimum Description Length

MITM..... Man In The Middle Attack

ML.....Machine Learning

MLP .....Multi-layer Perceptron

MOA ..... Massive Online Analysis

MSE.....Mean Square Error

NN .....Neural Network

OTP..... One Time Password

PRF..... Pseudo Random Function

RI .....Rule Induction

RIPPER.....Repeated Incremental Pruning to Produce Error Reduction

SEO .....Search Engine Optimization

SFH .....server form handler

SLD .....Second Level Domain

SMB.....Server Message Block

SMS ..... Short Messaging Service

SSH ..... Secure File Transfer Protocol

SSL ..... Secure Sockets Layer

SSNN ..... Self-Structuring Neural Network

SVM ..... Support Vector Machine

TANH ..... Hyperbolic Tangent Activation Function

TF-IDF ..... Term-Frequency-Inverse-Document-Frequency

TN ..... True Negative

TWF ..... Timed-Windowed Forgetting

UCI ..... University of California Irvine

TP ..... True Positive

VFDT ..... Very Fast Decsion Tree

VoIP ..... Voice over IP

WEKA ..... Waikato Environment for Knowledge Analysis

WMA ..... Weight Majority Algorithm

WOT ..... Web of Trust

List of Figures

List of Tables

Table of Contents

EVALUATING THE SSNN ALGORITHM

AN APPLICATION OF ESSNN TO PHISHING WEBSITES

CONCLUSIONS AND FUTURE WORK

CHAPTER 1

Introduction

1.1. Background and Motivation

With the digital revolution, digitized data have become easy to capture and to some extent inexpensive to store. The true value of raw data is substantiated based on its ability to produce useful information that might help in decision making or understanding the domain governing the data source. In most cases, data analysis is a conventional manual process where analysts would familiarise themselves with the domain, and with the help of statistical tools, they would generate reports and summaries to describe data insight. However, this approach may quickly deteriorate when the size and dimension of data increase. Hence, when the data exploration goes beyond abilities of typical manual analysis, analysts start looking for a more reliable knowledge discovery method to process data.

Data Mining (DM) comes into sight as an effective method to facilitate producing possibly useful knowledge for decision makers. DM or sometimes referred to as *Knowledge Discovery* (Kaufmann et al., 2011) is the method of analysing databases from different viewpoints and reforming them into meaningful forms. In other words, DM is the process of searching and analysing large amounts of raw data with the intention of discovering valuable rules and patterns (Linoff & Berry, 2011). A comprehensive definition of DM

comes from Gartner Inc[1] (Gartner IT Glossary, 2011) which defines DM as the process of analysing large amounts of data by means of some pattern recognition techniques, statistical, and mathematical practices in order to find out correlations, patterns and trends. However, the term DM can be generalized to any type of decision supporting system such as Business Intelligence (an umbrella term that refers to a variety of software applications used to analyse an organization's raw data), Machine Learning (ML) (a type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed), and Artificial Intelligence (AI) (the simulation of human intelligence processes by machines) (Usama et al., 1996).

Classification is a widely studied task in the DM. Classification is the process of developing a classifier (model) from a historical data set to forecast the value of the class variable(s) of an unseen example (Witten et al., 2011). Classification is a supervised learning approach because it is driven by means of an initial data set where the training examples are provided along with their corresponding class values. Typically, a classification model learns concepts (rules and patterns) from a static data set where the set of input features do not change. This data set is therefore assumed to contain all information required to learn the relevant knowledge pertaining to the underlying domain. A classification model created from a static data set is commonly called an offline classification model because the learning phase is a one shot process (Gaber, 2012). Such learning strategy, however, has proven unrealistic for many real world scenarios (Farid et al., 2013) (Hoens et al., 2012) (Polikar et al., 2001) such as phishing websites classification (fake websites created to induce users to voluntarily reveal their personal information) where the data set is often obtained over time in streams of instances instead of all training data sets being

[1] An information technology research and advisory firm.

available from the start. In addition, this learning strategy cannot accommodate new knowledge as soon as new data sets become available.

Evolving data sets may result in a system working in a dynamic (non-stationary) environment, which in turn raises the so-called concept drift dilemma. Concept drift signifies the changes in the relations connecting the input variables to the output variables. Concept drift has been discussed in detail in Chapter 2 including how the dynamic domains affected by the concept drift, why concept drift, and types of concept drift. Also in Chapter 3, we elaborate why the concept drift might occur in phishing websites and we gave some examples of the concept drift that occurring in phishing websites.

In general, learning under the presence of a continuously changing data set requires a model that can be updated regularly in order to leverage the newly collected data, while simultaneously maintaining the performance of the model on old data. The competing motivations of this aim give rise to another dilemma; that is the stability (maintain previously learnt knowledge) plasticity (learn new knowledge) dilemma (Sections 2.7 and 2.8). Among the most popular learning approaches that can effectively handle concept drift dilemma is the ensemble learning approach (Farid et al., 2013) (Wang et al., 2003) (Polikar et al., 2001) (Section 2.10.2). Such approach affords that the classification model learns new knowledge and at the same time preserves the previously learnt knowledge. Hence, the ensemble learning approach not only handles the concept drift dilemma, but also furnishes a balance between stability and plasticity.

Phishing websites is considered a typical example of the dynamic classification scenarios where the data set examples are continuously arriving (Ma et al., 2009) (Basnet et al., 2012). Phishing websites attracted researchers to address this problem from an intelligent point of view. In the last decade, several anti-

phishing techniques have been introduced as shown in Chapter 3. Yet, these techniques are far from perfect. For instance, whitelist and blacklist based techniques could not detect the zero-days phishing webpages. By contrast, intelligent methods based around DM have a possibility to recognize these websites (Aburrous et al., 2010 B). Nevertheless, most of the contemporary DM based anti-phishing models address the phishing websites as a static problem where the complete data sets are introduced to the DM algorithm (Abu-Nimeh et al., 2007), (Zhang et al., 2007), (Fette et al., 2007), (Miyamoto et al., 2008), (Basnet et al., 2008), (Aburrous et al., 2010 C), and (Abdelhamid et al., 2014). However, phishing websites is an evolving problem whereby the set of features that might be used to determine the type of a website are constantly changing over time (Basnet et al., 2012). Such a situation commonly referred to as a virtual concept drift (Polikar et al., 2001). Phishers know that the longer a phishing campaign uses the same set of features the more likely the anti-phishing parties will detect and deploy countermeasures against it. Phishers, therefore, adapt by constantly changing the set of features used to design such fake websites. Hence, a virtual concept drift might occur.

Yet, the virtual concept drift that occurs in phishing websites is considered a cyclical concept drift in the sense that the set of features that are used in predicting phishing might disappear at specific time and return to reappear again in the future (Ma et al., 2009) (Basnet et al., 2012). In addition, it is not guaranteed that all phishing website from an old campaign are no longer exist; and some of them are still alive. Hence, learning a new classifier from scratch might protect users from a phishing website generated from a new campaign but it might leave them susceptible to a phishing website generated from an old campaign as a direct result of the so-called catastrophic forgetting (Section 2.8). Thus, Catastrophic forgetting is another issue that we aim to address in our

thesis by ensuring that learning a new classifier does not mean forgetting the previously learned knowledge.

In general, building a classification model that is updated regularly in order to keep abreast of changes that may affect the model performance is an important and timely issue. The model should provide a high true positive (the proportion of legitimate websites that are correctly classified as such) and true negative (the proportion of phishing websites correctly classified as such) rates. The motivation behind the current research is to propose a new framework that is capable to create a classification model that can obtain knowledge from evolving data sets where a concept drift, particularly virtual concept drift might occur. Then, the proposed framework is applied to phishing websites classification problem.

1.2. Research Aims and Objectives

This thesis explores the area of classification and focuses on the domains that are working in dynamic (non-stationary) environments where a virtual concept drift might occur with the aim of creating a framework that can improve the performance of the classification models if a virtual concept drift has occurred. The framework will be then applied to a serious web security problem, which is the phishing websites classification problem in order to assess whether the framework can handle the virtual concept drift that characterizes the phishing websites.

Another aim of applying the proposed framework to phishing websites classification problem is to assess if the proposed framework will handle the catastrophic forgetting dilemma. That might be achieved by ensuring that the framework learns incrementally, hence, it will offer a balance between stability and plasticity.

The virtual concept drift that occurs in phishing websites has a peculiarity, as it might be guided by some malevolent intelligent agent rather than occurring naturally. Therefore, another aim of this thesis is to minimize the cause of intentional virtual concept drift by making use of the black box nature of Neural Network (NN). The black box nature of NN signifies that the only visible parts in any NN classification model are the input and output, whereas the process that transforms the inputs into outputs is obscured. This characteristic makes the task of picking a new set of features that can circumvent the classification model increasingly difficult. Nevertheless, most NN classification models are traditionally created using the trial and error method. Thus, one more aim of this thesis is to create an algorithm that simplifies structuring NN classifiers. The algorithm plays an important role in the proposed framework since it will derive the classifiers that are added to the ensemble. After confirming the presence of a concept drift, a new classifier will be created using the algorithm. Such a classifier is added to the previously derived classifiers forming an ensemble of classifiers each of which is considered an expert in a particular part of input features.

In order to apply the proposed framework to phishing websites, a set of variables known as inputs or features should be clearly identified. The set of features utilised in previous studies will be adopted in this research. However, we believe that we can find some more features if we dig into a set of phishing and legitimate websites. Hence, a set of phishing and legitimate websites will be collected, and then we will compare the phishing websites against the legitimate ones with the aim to come across new and possibly effective features.

1.3. Thesis Contributions

Several achievements have been accomplished in this research including the development of a new constructive NN algorithm that simplifies structuring binary NN based classifiers. Such an algorithm has been utilised in developing a classification framework based on ensemble self-structuring neural network. The framework is considered another contribution achieved in this research. The proposed framework has been applied to phishing websites classification problem. Hereunder, we underline the main contributions of this research.

1. A Classification Framework based on Ensemble Self-Structuring Neural Network (ESSNN)

A framework that is able to contend with the virtual concept drift is proposed in this research. As soon as a new data set arrives, and after confirming the presence of a virtual concept drift, a new classifier is created. The newly derived classifier is combined with all previously created classifiers to produce an ensemble of classifiers. When a new unseen example arrives, each classifier makes its calculations and gives the result about the value assigned to the class variable. The results produced from each classifier will be combined together; hence, the decision of ESSNN is a collective decision. The proposed framework learns incrementally and is able to accommodate new data and does not forget the previously learnt knowledge.

The proposed framework has been published in (Mohammad et al., 2016-A).

2. An Implementation of ESSNN to Phishing Websites

Most current DM based anti-phishing models are devoted to an offline learning strategy according to which as soon as the model is created it can gain no further knowledge. However, any phishing websites classification model is prone to the presence of a virtual concept drift. The proposed ESSNN has been

applied to phishing websites classification problem in order to assess whether it can handle the virtual concept drift that occurring in phishing websites. Another aim of applying the ESSNN to phishing websites is to empirically evaluate the performance of the ESSNN and verify that it will offer a balance between stability and plasticity when applied to phishing websites.

The results of applying the ESSNN on phishing websites have been published in (Mohammad et al., 2016-A).

3. An Improved Neural Network Structuring Algorithm

A new algorithm for structuring NN classifiers is created in this research. Such an algorithm is called Self-Structuring Neural Network algorithm (SSNN), which facilitates the creation of NN classifiers. This can simplify deriving NN classifiers with good generalization ability. This algorithm does not need to guess an appropriate NN structure; it automatically finds it. The SSNN algorithm creates the classifiers that are added to the ESSNN.

The SSNN algorithm has been published in (Mohammad et al., 2013-B).

4. Empirical Evaluation of SSNN

Several experiments have been conducted in order to evaluate the performance of SSNN algorithm when compared to several other classification algorithms including decision tree, logistic regression, and standard feed-forward neural network algorithm implemented in WEKA (Hall et al., 2011). Two sets of experiments have been conducted. In the first set, ten different binary data sets from University of California Irvine repository (UCI Repository) have been used. However, in the second set of experiments, the SSNN has been applied to phishing websites classification problem. The experimental results show that the SSNN algorithm is able to produce classifiers with good generalization ability.

The experimental results have been published in (Mohammad et al., 2016-B).

5. Dissemination of Structured Phishing Websites Training Data sets

One of the key challenges encountered in this research has been the unavailability of a structured training data set. In fact, this challenge faces any researcher in the field. Although plenty of articles about predicting phishing websites have been published, no structured training data sets have been offered publicly. To the best of our knowledge, we have introduced the first publicly published structured phishing websites training data sets.

These data sets can be accessed from the University of Huddersfield repository (Mohammad et al., 2015-C) and from the University of California Irvine repository (UCI Repository) (Mohammad et al., 2015-B). Our data sets have been utilised in several Journal articles, Conference articles and PhD theses as in (Abdelhamid, 2013) (Singh & Patil, 2014) (Abdelhamid et al., 2014) (Zeydan et al., 2014) (Abdelhamid, 2015) (Qabajeh & Thabtah, 2014) and (Mansour & Alshihri, 2015).

6. Background and Literature Review on Phishing

Nowadays, phishing websites problem has attracted many researchers. In this thesis, we have produced a detailed survey of phishing websites so that other scholars can use our analysis as a starting point for their researches. In this survey, we shed light on the current developments in phishing and provide a comprehensive study on the precautionary measures with emphasis on the intelligent anti-phishing methods. We approach the phishing websites problem from a different perspective, since in this thesis the problem has been presented as a dynamic not static problem. Such kind of problems requires specific attention to some criteria such as concept drift, catastrophic forgetting and

stability-plasticity. These criteria have also been discussed in this thesis. The ensemble approach poses as one possible solution that can address the dynamic nature of the problem.

Parts of this review have been published in (Mohammad et al., 2012) (Mohammad et al., 2013-A) (Mohammad et al., 2013-B) (Mohammad et al., 2013-C) (Mohammad et al., 2016-A) (Mohammad et al., 2016-B) and an extended full version in (Mohammad et al., 2015-A).

1.4. Thesis Outline

The subsequent chapters of the thesis are organised as follows:

Chapter 2 elaborates the main difficulties of creating any classification model. Concept drift, catastrophic forgetting, and stability plasticity dilemmas have been debated as the main issues to be considered when creating any classification model working in a dynamic domain. The difficulties of creating neural network classifiers are also discussed in this chapter. In addition, this chapter presents a brief overview of some offline classification algorithms.

Chapter 3 discusses the phishing phenomenon in detail and shows that phishing websites is a dynamic classification problem where a virtual concept drift might occur. In addition, in this chapter we will review and evaluate the contemporary researches on the intelligent anti-phishing measures in order to recognise the issues that is still predominating this area. Also in this chapter, we argue that an adaptive intelligent precaution model is needed to cope with the evolving nature of the phishing websites problem.

Chapter 4 describes the main phases of the Ensemble Self-Structuring Neural Network (ESSNN) framework that continuously gains knowledge from evolving data sets. In addition, in this chapter we have proposed an improved NN structuring algorithm called Self-Structuring Neural Network (SSNN) that is considered the cornerstone in creating the proposed classification framework.

Chapter 5 evaluates the performance of the SSNN algorithm. Several experiments have been accomplished to assess the performance of the SSNN algorithm. Two sets of experiments have been done. In the first set, the algorithm has been assessed against several DM classification algorithms on a number of binary data sets from UCI repository. In the second set, the SSNN algorithm has been applied to phishing websites classification problem and the results have been compared with other DM classification algorithms. Also in this chapter, we shed light on the important features that have proved to be sound and effective in predicting phishing websites.

Chapter 6 evaluates the applicability of the ESSNN when applied to phishing websites. Several experiments have been conducted to compare the performance of the ESSNN with other DM techniques. Finally, the experimental results are discussed.

Chapter 7 summarizes what have been achieved in this research. In addition, in this chapter several future works are presented.

CHAPTER 2

Classification and Dynamic Domains

2.1. Introduction

The function type that a DM tool provides is usually referred to as a DM method (Witten et al., 2011). The most popular and commonly used DM methods are clustering, prediction, association rule discovery, and outlier analysis (Chen et al., 1996) (Linoff & Berry, 2011). Nevertheless, the non-inclusion of a method in the first place list does not mean that this method does not exist, but that could be because some researchers assign special terms for the method. For instance, in case of prediction, if the class variable holds categorical values; then it is called classification. On the other hand, if the class variable holds real values; then the prediction process is called regression (Fayyad et al., 1996). There is no specific DM method suitable to address all problems. As soon as it is time to select a DM method for a specific problem, the selection should be taken very carefully, as one method might fit a specific domain, but it might produce poor results somewhere else. In general, the DM methods can be divided into two categories namely predictive (supervised), and descriptive (unsupervised) (Witten et al., 2011) (Chen et al., 1996). Supervised DM methods produce models that are able to forecast a target variable (Witten et al., 2011). However, unsupervised DM methods divide the observations into clusters where similar observations are grouped together (Witten et al., 2011).

Classification is a well-known task in data mining. A classification model aims to map a data set item to one of predefined categorical groups.

The most commonly used classification algorithms are (Witten et al., 2011), Decision trees (DT) (Quinlan, 1979) (Quinlan, 1986) (Quinlan, 1993) (Quinlan, 1998), Rule Induction (RI) (Cohen, 1995), Bayesian Network (BN) (Friedman et al., 1997), Support Vector Machine (SVM) (Witten et al., 2011) (Cortes & Vapnik, 1995), and Logistic Regression (LR) (Witten et al., 2011). These algorithms are commonly used in the domain of phishing classification (Abu-Nimeh et al., 2007), (Zhang et al., 2007), (Fette et al., 2007), (Miyamoto et al., 2008), (Basnet et al., 2008), (Aburrous et al., 2010 C), and (Abdelhamid et al., 2014). Such algorithms have been briefly discussed in Appendix A.

However, in order to produce a classification model that can achieve good classification performance several criteria must be put in mind. For instance, in dynamic domains the training data set is not static, but steadily evolving. Such data set might make the previously discovered rules and patterns become inaccurate or outdated. Hence, a model that is considered perfect might produce poor results over time. Another problem that should be addressed when creating any classification model is the overfitting problem. Overfitting occurs when the training phase runs for a long time with the aim of decreasing the error-rate on the training data set, but the overall performance on the testing data set may deteriorate. Corrupted data set, i.e. a data set that contains noisy, missing, or irrelevant items is also an important issue when creating any classification model.

In this chapter, we elaborate the main difficulties that may affect the overall performance of any classification model, with emphasis on classification models that are working in dynamic domains. Concept drift, catastrophic forgetting, and stability plasticity dilemmas will be discussed as the key issues

to be addressed when creating any classification model. Neural network structuring approaches and the difficulties of creating a neural network classification models are also discussed in this chapter.

2.2. Continuity Strategies for Classification models that are Working in Dynamic Domains

As soon as a classification model is installed and utilised in everyday operations, its performance should be assessed regularly. If there is no significant change in the future performance, keeping the initial model might be justified. If, on the other hand, the performance drops significantly, keeping the model might lead to unstable results as an effect to the incorrect decisions being made. In general, two possible strategies could be followed when applying any classification model:

1. Keep applying the model without making any improvements on it even if some changes occur in the domain. This strategy is computationally cheap but with the risk that the model will be susceptible to concept drifts.
2. The examination and alteration of the existing model to reconstitute it in a new form to cope with any changes occurring in the domain. Although this strategy might be expensive, it ensures the availability of an up to date model. Three possible strategies might be applied in this case:
   A) Re-engineering the current model by tuning a set of its parameters.
   B) Retraining the model as soon as a new training data set becomes available.
   C) Moving from one training method to another because the new training method might be more suitable to process the new data set.

One may say that option (B) above is the best choice. However, offline classification algorithms are not by default incremental learners since the previously learnt knowledge will be lost after a new model is created (Franklin et al., 2007) (Polikar et al., 2001).

2.3. Concept Drift and Dynamic Domains

Normally, offline classification algorithms attempt to learn knowledge from a static training data set. However, this approach has proven impractical for many scenarios where a concept drift may emerge (Gaber et al., 2005) (Tsai et al., 2009) (Masud et al., 2012) (Hoens et al., 2012) (Farid et al., 2013). When designing a classification model, a series of training data set examples can be obtained occasionally. However, if there is any alteration in the underlying data between any two consecutive time steps, there is a concept drift resulting in a model working in a non-stationary domain.

In general, the concept drift signifies that the relations connecting the input variables to the output variables are changing over time in unpredicted ways (Widmer & Kubat, 1996) (Tsai et al., 2009) (Hoens et al., 2012) (Farid et al., 2013) (Gama et al., 2014). In other words, concept drift can be described as the changes that occur in the learned knowledge. This results in some problems because the predictions become less accurate as the time passes. Formally, concept drift can be defined as follows (Gama et al., 2014):

$$d_{t0}(F, r) \neq d_{t1}(F, r)$$

Where d signifies the joint distribution at time t_0 and t_1 between the set of input variables F and the target variable r.

2.4. Why Concept Drift

If we look at the big image of any supervised training data set, we can see that every training example consists of only two parts, features part and class part. Any change occurs in any part leads to the emergence of the concept drift. Hence, we can distinguish between two styles of concept drifts (Parikh & Polikar, 2007):

1. Real concept drift: refers to changes in the class variable, i.e. the emergence of some new values for the class variable. Such changes can happen either with or without change in the input variables (input features).
2. Virtual concept drift: refers to changes in the input features, i.e. the emergence of some new input features. Such change happens without any changes in the class variable (Polikar et al., 2001).

For dynamic domains where a virtual concept drift occurs, irrelevant or redundant features might be restricted to a specific period of time. However, irrelevant set of features might be ignored permanently when creating an offline classification model even if these features become relevant in the future. From a practical point of view, it does not matter what kind of concept drifts occur, as in all cases the model needs to be updated.

2.5. Types of Concept Drift

Concept drift may manifest in different forms as per Figure 2.1.

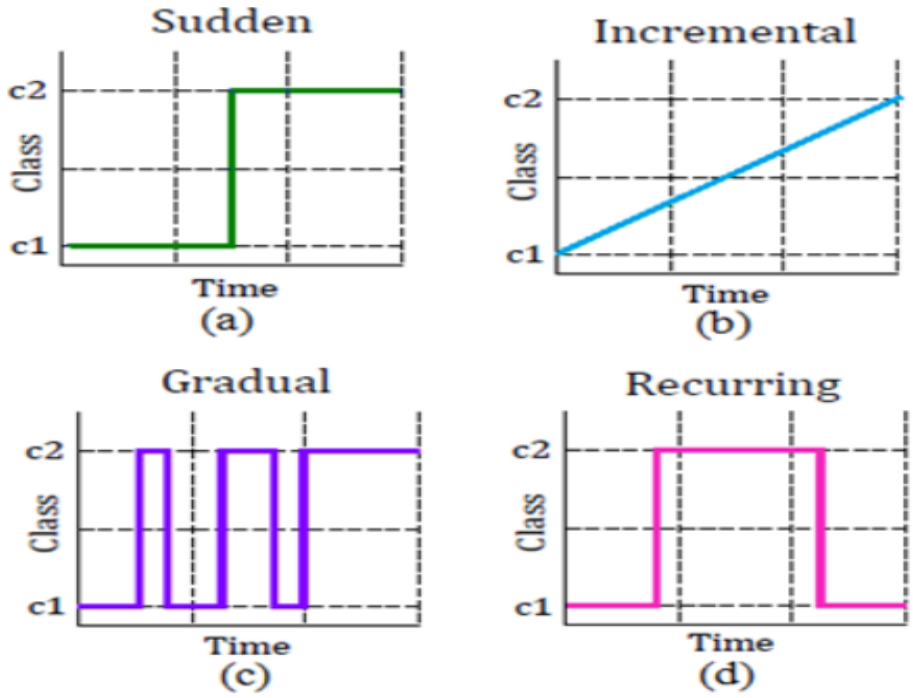


Figure 2.1 Concept drift categories

Concept drift may occur suddenly by switching from one concept to another (e.g. replacement of a sensor with another sensor that has a different calibration in a chemical plant) (Gama et al., 2014), or incrementally, by passing through several concepts to reach the final one (e.g. a sensor slowly wears off and becomes less accurate) (Gama et al., 2014). However, incremental concept drift can be seen as small sudden drifts (Gaber, 2012). Some researchers may use gradual concept drift as a synonym for incremental concept drift (Farid et al., 2013). The last category of concept drift is recurring or cyclical concept drift (e.g. seasonal fashion). This kind of drift arises (in case of virtual concept drift) when some features underlying the class variable switch over time at irregular time intervals (e.g. consider a text stream where each data point is a document, and each word is a feature. Since it is impossible to know which words will appear in the future, the complete feature space is unknown). Therefore, some influential set of features may disappear for a while, but they may return and reappear again in the future. However, it is not clear when the previous features may appear again.

2.6. Offline, Online, and Incremental Learning strategies

Typically, classification models are trained in an offline mode. Firstly, a classification model learns how to perform a certain task and later it is applied to perform similar tasks. No tasks can be performed during the learning phase, and as soon as the learning phase is finished, the model can no further be adjusted or modified. In contrast, online and incremental learning approaches have been employed with various meanings in the literature and are frequently mixed. The most essential difference of online and incremental learning from offline learning is that online and incremental learning approaches assume that the training data set examples emerge continuously over time. Hence, the

training data set is not static, but steadily evolving. Online learning approaches are able to perform a certain task and learn at the same time. These approaches are updated whenever a new training instance is offered; being capable to achieve lifelong learning since they do not ever stop learning. On the other hand, incremental learning approaches can also work in the changing domains. Yet, incremental learning approaches are capable to work in a more general scenario, where the training data sets can be processed in chunks (Oza & Russell, 2001-A). In other words, incremental learning approaches can process new data in terms of blocks of training data sets, whereas online learning approaches process every training instance separately as soon as it arrives. This differentiation is particularly important in this research because the framework proposed in this study is considered an incremental learning method that processes each training chunk as soon as it becomes available.

2.7. The Stability-Plasticity Dilemma

Any classification model works in a dynamic domain has to be improved constantly to contend with any changes that may affect its performance. To achieve this goal, the training procedure has to be able to cope with the plasticity of the new knowledge. However, learning new knowledge should not mean the loss of the old one. Yet, learning new information without losing previously learnt knowledge raises the so-called stability-plasticity dilemma (Hoens et al., 2012). The dilemma draws attention to the fact that completely stable models will preserve previously learnt knowledge, but will not accommodate any new knowledge. In contrast, completely plastic models will learn new knowledge, but will not preserve previously learnt knowledge.

2.8. Catastrophic Forgetting

Several DM models and algorithms are faced with the so-called Catastrophic Forgetting dilemma. Catastrophic forgetting, or sometimes called "Catastrophic Interference" or "The Sequential Learning Problem" is first introduced by (McCloskey & Cohen, 1989). Catastrophic forgetting occurs when a model is trained on one task, and then trained on a second task; the model might forget how to do the first task (Goodfellow et al., 2015). In other words, catastrophic forgetting happens when old knowledge is deleted when trying to obtain new knowledge.

Catastrophic forgetting is normally seen in conjunction with the so-called stability-plasticity dilemma (Hoens et al., 2012). Normally, catastrophic forgetting and the stability-plasticity are tackled by ensuring that the DM model learns incrementally. Human learning is the best example of the incremental learning. One learns the concept description from the offered information and incrementally refines these descriptions when new information and observations become available. Newly added information is used to improve knowledge structure and rarely causes reformulation of all the knowledge the person has about the subject at hand. Incremental learning is an important capability for brain-like intelligent systems.

2.9. Viability of Incremental Learning for Dynamic Domains

Faced with the ever-evolving data sets, the direction of adaptive DM approaches seems to be the right choice to reveal the new knowledge that might be included in such data sets. This knowledge if not discovered might affect the overall performance of the classification model. Any adaptive model is subject to progressive developments. How good the model is in describing the data set insight controls the direction and rate of these developments.

Normally, an adaptive model can be achieved by ensuring that the classification model learns incrementally (Farid et al., 2013). In general, an incremental system should satisfy the following criteria (Polikar et al., 2001):

1. Learn new knowledge from any new data set that becomes available
2. Learning new knowledge does not lead to forgetting previously learnt knowledge
3. Learning new knowledge does not involve accessing previous training data sets
4. Should be able to accommodate new classes if introduced in the new data set

However, the fourth criterion is not applicable for domains where a virtual concept drift might occur. Incremental learning is normally a reaction to the presence of a concept drift and it can be described in several ways, such as online versus batch methods subject to the number of instances used at each training step; or single versus ensemble based classifiers subject to the number of classifiers used to come up with the final decision. Several techniques have already been suggested with view to creating incremental models that allow the learning of new knowledge without forgetting old ones.

The simplest technique is called interleaved learning (Seipone & Bullinaria, 2005) where the original training data set is combined with the new one. Then we can either retrain the existing model, or discard the previous model and build a new one from scratch. This way, the interleaved learning technique fulfils the first and the second criteria above, but the third criterion is clearly violated. Moreover, this technique might be computationally expensive and memory consuming (Parikh & Polikar, 2007). Some other techniques work by reducing the size of the full data set by utilising a subset of the new examples rather than using the complete data set (Engelbrecht & Brits, 2001). Yet, this approach still needs access to the old data. Several methods use some form of a

sliding window over the incoming data. The batch of examples that fall inside the window is considered the new training data set, and a new classifier is created from such batch. However, sliding window technique may result in the so-called catastrophic forgetting (Hamker, 2001), hence, infringing the second criterion of the incremental learning definition. Moreover, this technique may fail if the concept drift in the problem is a cyclical concept drift. However, the use of ensemble methods is popular in incremental learning scenarios due in part to their empirical effectiveness (Polikar et al., 2001).

In this research, the ensemble based incremental learning method has been utilised to create an adaptive classification framework.

2.10. Handling Catastrophic Forgetting and Stability Plasticity Dilemma

Two different approaches can be followed in order to produce a classification model that offers a balance between stability and plasticity those are, single based approach and ensemble based approach. These approaches ensure that the classification model learns incrementally and is not susceptible to catastrophic forgetting (Hamker, 2001).

2.10.1. Single Classifier based Approach

In this sense, a single classifier is updated regularly to ensure that it will efficiently contend with the stability-plasticity dilemma. This approach selects and maintains a portion of the earlier training examples that are then merged with new examples in subsequent training phases. This approach is also known as window based technique (Widmer & Kubat, 1996). Window based technique provides a simple forgetting procedure. This technique stores only the most recent examples in the first-in-first-out (FIFO) data structure. As soon as new instances arrive, they are embedded into the start of the window. A same

number of instances are expelled from the end of the window as indicated in Figure 2.2.

Figure 2.2 Window based learning schema

This technique assumes that the knowledge learnt from the older instances are conceptually worthless for classifying new examples and therefore such old instances are dropped from the training data set. A classifier is created using the data set examples within a fixed or a dynamic sliding window. The fixed sliding window techniques store the most recent n instances, where n is specified by the system designer. At each training phase, the learning algorithm builds a new classifier using the examples within the window. However, the examples that are moving out of the window are discarded. Yet, in the dynamic sliding window, the window size is dynamically adjusted over time (Gama et al., 2014). The key challenge in the dynamic sliding window technique is to select an appropriate window size. A small window reflects the current distribution more accurately but that may affect the classifier performance on the old examples (Gama et al., 2014) because the old examples might be dropped from the

window as there is no space to keep them within the window. On the other hand, a large window might preserve the classifier performance on the older examples, but it reacts slowly to the concept drift (Gama et al., 2014). Examples of window based techniques include, FLOating Rough Approximation (FLORA) family of algorithms (Widmer & Kubat, 1996), Floating Rough Approximation in Neural Network (FRANN) (Kubat & Widmer, 1995), Timed-Windowed Forgetting (TWF) (Salganicoff, 1997), and ADaptive sliding Window (ADWIN) (Bifet & Gavaldà, 2006).

Weight based approach is considered a special case of the window based technique. Traditionally, the age of an example determines whether it will be dropped or maintained within the window. However, some old examples may still valid for the current state. Hence, the key issue in the weight based windowing method is how to choose the most relevant examples from the old data set, maintain and merge them with the most recent data set, and use them in upcoming learning episodes. Weight based method gives an advantage over windowing method because it allows the learning algorithm to have more control over how examples are incorporated into the model than simply exist or not exist (Hoens et al., 2012). Each data set example is assigned a weight. The weights determine which data set instance becomes outdated and then drop it from the training data set. However, assigning a weight for each data set example might consume time (Gama et al., 2014) and some data set examples that are dropped from the training data set may become relevant again in the future. An example of such algorithms is WINNOW algorithm (Littlestone, 1988).

In general, the window based approach achieves good results in small data sets (Gaber et al., 2005). In addition, this approach might bias the classifier toward the recent state (Gaber et al., 2005) (Gama et al., 2014). Most importantly, this

approach might not be the right choice if we face a cyclical concept drift (Gama et al., 2014). Phishing websites problem is a typical example where the cyclical concept drift might occur.

Information theoretic measures, such as entropy (Coifman, 1992), mutual information (Peng et al., 2005), or hoeffding bounds (Hoeffding, 1963) have also been used for updating a classifier, typically using the decision tree algorithm. The Very Fast Decision Tree algorithm (VFDT) (Domingos & Hulten, 2000) is one of the first algorithms that use hoeffding bounds to grow decision trees in streaming data. The authors in (Domingos & Hulten, 2000) reveal that applying hoeffding bounds to a subset of data (in case of continuously evolving data sets) can with high confidence choose the same split attribute as when using the whole training data set. Many modifications to the VFDT algorithm have been made. One of which is the Concept Adapting Very Fast Decision Tree (CVFDT) (Hulten et al., 2001). CVFDT uses a fixed-size window to determine which nodes might need updating. More recently, a new algorithm has been proposed that improves the CVFDT. Such algorithm is called Hoeffding Adaptive Tree (HAT) (Bifet & Gavaldà, 2009). In this algorithm, each node in the tree can determine which of the previous examples are relevant for it.

2.10.2. Ensemble based Approach

This approach combines a set of classifiers whose individual decisions are combined together in some way with the aim of producing an improved composite model with high classification performance. Majority voting is considerably the simplest method for combining classifiers outcomes (Lam & Suen, 1997). Each classifier in the ensemble makes its own decision on the value of the class variable for a specific instance. The final decision is assigned to the class value that has a consensus or when at least d of the classifiers are agreed

on the class value. *d* is calculated as per equation 2.1 where *n* is the number of classifiers in the ensemble.

$$d = \begin{cases} \frac{n}{2} + 1 \text{ if n is even} \\ \frac{n+1}{2} \text{ if n is odd} \end{cases} \quad (2.1)$$

The Weighted Majority Algorithm (WMA) (Littlestone & Warmuth, 1994) is another technique for combining classifiers outcomes. In this method, a set of classifiers are weighted according to their performance. For each testing example, the ensemble collects weighted votes from all classifiers in the ensemble and produces the prediction that has a higher vote. If the ensemble makes a mistake, the weight of the classifiers that contributed to the wrong prediction is reduced by a certain ratio.

Naturally, ensemble based approach learns incrementally, does not forget the previously learnt knowledge and does not require access to the previously seen training data sets (Polikar et al., 2001). Ensemble based methods have been the most common solution for domains where a cyclical concept drift occurs (Hoens et al., 2012) since an ensemble normally contains classifiers built from past data set; such classifiers can be reused to classify newly arrived examples if they are drawn from a reoccurring concept.

When creating an ensemble, it is important to have a set of classifiers that are competent and at the same time complement to each other (Gama et al., 2014) hence, if one classifier makes an error, the others will correct that error. In (Hansen & Salamon, 1990) the authors proved that one key factor when creating an ensemble is to have a set of classifiers each of which produces an error rate < 50%.

Several techniques might be used to create a set of classifiers that are un-correlated as much as possible. For instance, Bagging (Breiman, 1996) and

Boosting (Freund & Schapire, 1997) deliver classifiers diversity by training each classifier with a different subset of training examples. Bagging utilises resampling with replacement technique to produce different training data sets. Nevertheless, some examples may be selected several times while others may not be selected at all. On the other hand, in Boosting, instances that are incorrectly classified by previous classifiers are selected more often than instances that are correctly classified. Bagging and Boosting techniques assume that the training data set is completely offered at the training phase (Grbovic & Vucetic, 2011) hence, diversity can be easily drawn by splitting the original data set into different subsets. However, the framework proposed in this research is more inclined towards dynamic nature where the data set examples are obtained on different time slots. Other techniques ensure diversity by creating several classifiers each of which is produced by means of a different DM algorithm.

Ensemble approach has successfully been applied in several domains, for example, intrusion detection (Mukkamala et al., 2003), anomaly detection (Shoemaker & Hall, 2011), spam detection (Wang et al., 2009), cyber-attack classification (Dharamkar & Singh, 2014) and credit card fraud detection (Wang et al., 2004).

2.11. Issues when Structuring a Neural Network Model

ANN proved its merits in several classification domains. A NN classifier is considered as a black box, i.e. the only visible parts are the input and output, whereas the process that transforms the inputs into outputs is concealed. This characteristic has successfully applied in different security domains where the intentional concept drift might occur (Arvandi et al., 2008), (Alallayah et al., 2010), and (Al-Ubaidy, 2004). This thesis has benefited from the black box nature in order to minimize the cause of intentional virtual concept drift.

Nevertheless, NN classifiers are normally created using a painstaking trial and error method. Although selecting a suitable number of hidden neurons and determining the value of some parameters, i.e. learning rate, number of hidden layers, and epoch size showed to be crucial when constructing any NN classification model (McCaffrey, 2012) there is no clear mechanism for determining such parameters, and most model designers rely on trial and error approach. However, although the trial and error might be suitable for domains where rich prior experience and a knowledgeable NN expert exist, it often involves a tedious process because rich prior knowledge and experienced human experts are hard to get in practice. Moreover, the trial and error technique has been criticised of being a time-consuming process (Ma & Khorasani, 2003).

A poorly structured NN model may under-fit the training data set. On the contrary, exaggeration in re-structuring the model to suit every single instance in the training data set might cause the model to be over-fitted (Polikar et al., 2001). One possible solution to avoid the overfitting problem is by restructuring the NN model in terms of tuning certain parameters, adding new neurons to the hidden layer or adding a new layer to the network (Widrow et al., 1990) (Kwok & Yeung, 1997). A NN with a small number of hidden neurons may not have a satisfactory representational power to model the complexity in the training data. On the other hand, a network with too many hidden nodes could over-fit the data. However, at a certain stage, the model can no further be improved; hence, the structuring process should be terminated. Therefore, an acceptable error margin should be specified when building any NN model, which itself is considered a problem, since it is difficult to specify the desired error rate a priori. For instance, the user may set the desired error-rate to an unreachable value, which may led the model to be trapped in a local minima

(Kriesel, 2007) as shown in Figure 2.3 or sometimes the model designer may set the acceptable error-rate to a value that can further be improved.

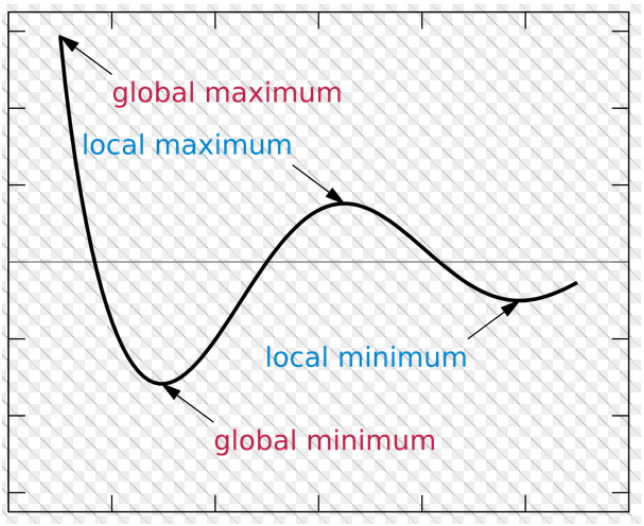


Figure 2.3 Local and Global minimum/maximum

Overall, automating the structuring process of NN models is a timely issue and it might displace some of the burden from the model designer. However, automating structuring NN models does not merely means adding new neuron(s) to the hidden layer because the more hidden neurons does not necessarily mean that the accuracy will improve (Ganatra et al., 2011) (Hornik et al., 1989). Hence, it is important to try improving the NN performance by adjusting several parameters such as the desired error-rate and the learning rate before adding a new neuron to the hidden layer. Sadly, selecting the learning rate value is also a trial and error process. Yet, although several studies have been made to come up with the best NN structure, the optimal desired error-rate and learning rate values are still concealed.

2.12. Neural Networks Structuring Approaches

2.12.1. Constructive Approach

This approach starts with a simple NN structure, i.e. one hidden layered NN with a single neuron in the hidden layer (Islam et al., 2009) and recursively new parameters, i.e. hidden layers, hidden neurons, and connections are added to the initial structure until reaching a satisfactory result. After each addition, the entire network or only the recently added parameter is retrained. Constructive approach is relatively easy for inexperienced users because they are normally asked to specify few initial parameters, for example the number of neurons in the input layer and epoch size and then new parameters are added to the network. This approach is computationally efficient because it searches for small structures first (Islam et al., 2009). However, constructive approach has some hurdles that should carefully be addressed. For example, the user has to decide when to add a new hidden neuron, when to stop the addition process, and when to terminate training and produces the network (Kwok & Yeung, 1997).

2.12.2. Pruning Approach

Unlike constructive approach, the pruning approach starts with an oversized NN structure, i.e. a multi hidden layered NN with a large number of hidden neurons in each hidden layer. Later on, some parameters, i.e. connections, hidden neurons, and hidden layers are removed from the network. After each training process, the user removes some parameters from the network and the new structure is retrained so that the remaining parameters can compensate the functions played by the removed parameters. If the network performance improved, the user removes more parameters and retrains the network again. However, if the network performance does not improved, the user restores

what have been deleted and tries to remove other parameters. This process is repeated recursively until achieving the final network. Usually, only one parameter is removed in each pruning phase (Islam et al., 2009). Overall, this approach is a time consuming. In addition, the user does not know a priori how big the initial NN structure should be for a specific problem.

2.12.3. Constructive-Pruning Approach

This approach comprises two phases, a constructive and a pruning. During the training phase new hidden layers, hidden neurons, and connections are added. The constructive phase may result in a ridiculously complicated structure. Hence, a pruning phase is employed to reduce the network structure and at the same time preserves or improves the network performance. The pruning phase can run simultaneously with the constructive phase, or it can be started as soon as the training phase is accomplished (Islam et al., 2009).

2.13. Feature Selection Methods

Features selection is an important step when creating any classification model. As shown in section 2.4, for domains where a virtual concept drift might occur, the set of input features change over time. Therefore, it is important to identify the set of significant features before proceeding in updating the classification model or creating a new one. In general, any classification model aims to approximate the functional link $f()$ between an input attributes $N = \{n_1, n_2, \ldots, n_W\}$ and an output attributes $T = \{t_1, t_2, \ldots, t_H\}$. Sometimes the output attributes can be concluded by only a sub-set of the input attributes $\{n_{(1)}, n_{(2)}, \ldots, n_{(w)}\}$ where $w < W$. Hence, it might be reasonable not to use the whole set of input attributes. Yet, with the availability of sufficient resources, it might be acceptable to use all input attributes, even the ones that are redundant

or irrelevant. Two different techniques can be used to select the most effective set of features those are as follows:

A) Filter method

In this approach, the selection process is independent of the data mining algorithm that will be utilised on the chosen features. Normally, filter methods evaluate the features significance by examining the inherent characteristics in the data. Once features relevance score is calculating the set of features that do not pass a pre-determined threshold are deleted. However, the set of remaining features is presented to the data mining algorithm as an input features. This technique is computationally fast and simple (Sánchez-Maroño et al., 2007). In addition, in this method the features selection phase needs to be completed one time only (Sánchez-Maroño et al., 2007).

Several feature selection algorithms can be used, for instance, Information Gain (Shannon, 1948), Chi-Square (Greenwood & Nikulin, 1996) and Gain Ratio (Quinlan, 1993).

1- Information Gain (IG): is the most frequently used algorithm in filter methods (Bramer, 2013), (Dash & Liu, 1997), (Yu & Liu, 2004). Information Gain employs an information theoretic measurement called entropy, which assesses the uncertainty in a data set associated with a particular variable (normally the class variable). The entropy is calculated as per equation 2.2.

$$E(D) = \sum_{i=1}^{N} -p_i \, log_2(p_i) \qquad (2.2)$$

Where p_i is the relative frequency of class i in data set D comprising N classes. In case of binary classification, we can customize the entropy equation as per equation 2.3.

$$E(D) = -P_l \ log_2(P_l) \ - \ P_p \ log_2 (P_p) \quad (2.3)$$

Where P_l signifies the possibility that a sample holds the first class; and P_p is the possibility that the sample holds the second class. After calculating the entropy, we start examining the effect of each feature on the IG. The feature that minimizes the entropy is added to the minimal data set. The IG is calculated as per equation 2.4.

$$IG\ (D, F) = E(D) - \sum_{v \in values(F)} \frac{Dv}{D} \ E(D_v) \quad (2.4)$$

Where $E(D)$ is the entropy of the whole data set, F is the feature in which the IG is assessed, and D_v is the number of features in D, and $E(D_v)$ is the entropy of the sub data set that has the value v for the feature F. The higher the IG value, the more helpful the feature will be for classification.

2- Chi-Square: is utilised to measure whether the occurrence of a specific class and the occurrence of a specific input attribute (feature) are independent. The high value of Chi-Square means that the class attribute and the input attribute are dependent, hence, the input feature is added to the selected feature set. On the other hand, low value of Chi-Square means that the input feature is independent of the class and therefore it is considered irrelevant for classification (Witten et al., 2011). Chi-Square is calculated as per equation 2.5.

$$x^2(f, t) = \frac{N(AD-CB\ )^2}{(A+C)(B+D)(A+B)(C+D)} \quad (2.5)$$

Where f is an input feature, t is a class variable, A denotes how many times that t and f co-occur, B is the number of times that f occurs without t, C is the number of times that t occurs without f, D is the number of times neither t or f occur, and N is the number of instances in the data set.

3- Gain Ratio: utilises an iterative process for feature selection. These iterations terminate when there is only predefined number of features remaining. The higher the Gain Ratio for a specific feature the more useful the feature is for classification. The Gain Ratio uses split information for normalizing the Information Gain score. The split information value represents the potential information generated by partitioning the training dataset *D* into *V* partitions, resulting to *V* outcomes on attribute *A*. Split information is calculate as per equation 2.6.

$$Split\ Info\ A(D) = -\sum_{j=1}^{v} \frac{|Dj|}{|D|} \times log_2 \frac{|Dj|}{|D|} \qquad (2.6)$$

The Gain Ratio is calculated as per equation (2.7)

$$Gain\ Ratio\ (A) = Information\ Gain\ (A) / Split\ Info\ (A) \qquad (2.7)$$

B) Wrapper method

This technique uses the results of the data mining algorithm to assess how good a given feature subset is. The main advantage of this method is that the quality of a features subset is assessed by the performance of the data mining algorithm applied to that subset. However, this technique is much slower than the filter method (Sánchez-Maroño et al., 2007). In addition, this method is computationally expensive compared to filter methods.

2.14. Chapter Summary

This chapter provides and overview of the main obstacles when creating any classification model. Concept drift, catastrophic forgetting, and stability plasticity dilemmas have been debated as the main issues to be considered when creating any classification model for dynamic. Two possible approaches that can be used to provide a balance between stability and plasticity when

creating any classification model for dynamic domains have been discussed, i.e. single classifier approach and ensemble based approach. In addition, we discussed the main issues that should be taken into account when creating NN based classification models. Also, in this chapter several classification algorithms have been briefly described.

CHAPTER 3

Phishing Websites and Contemporary Anti-Phishing Techniques

3.1. Introduction

The Internet has become an essential component of our everyday social and financial activities. The Internet is not important only for individual users, but also for organizations, because organizations that offer online trading can achieve a competitive edge by serving worldwide clients. The Internet facilitates reaching clients all over the world with effective use of e-commerce and without any market place restrictions. Therefore, the number of consumers who use the Internet to perform procurements is increasing significantly. Hundreds of millions of dollars are transferred through the Internet every day. This amount of money is enticing the fraudsters to carry out their fraudulent campaigns. Hence, Internet users may be susceptible to different types of web threats, which may cause financial damages, identity theft, loss of private information, brand reputation damage and loss of customers' confidence in e-commerce and online banking. Therefore, some users may doubt the suitability of the Internet for commercial transactions.

Phishing is considered a form of web threats that is defined as the art of impersonating a website of an honest enterprise aiming to obtain users' private information such as social security number, username, and password. Presumably, these fake webpages have high visual similarities to the genuine

ones. Technical tricks and social engineering are commonly combined together to start a phishing attack. Normally, a website phishing attack starts by sending an e-mail that seems authentic to potential victims urging them to update or validate their information by following a URL link within the email. Predicting and stopping phishing attacks is a critical issue towards protecting online transactions. The Internet community has devoted a considerable amount of effort into defensive measures against phishing. However, the problem is continuously evolving and ever more complicated deceptive methods are appearing. Therefore, a promising intelligent solution that must be improved constantly is needed to keep pace with this continuous evolution. Recognizing phishing websites accurately reflects how good an anti-phishing tool is.

In this chapter, the phishing phenomenon will be discussed in detail with the aim to realize the up to date developments in phishing and its precautionary measures. In addition, we produce a survey of the contemporary researches on the intelligent anti-phishing measures and provide an evaluation of these researches to recognise the gap that is still predominating this area. Also in this chapter, we argue that an adaptive intelligent precaution framework is needed to cope with the evolving nature of the phishing websites problem. Such a framework should address several issues such as, concept drift, catastrophic forgetting, and stability-plasticity dilemma.

3.2. Phishing Websites Timeline

In the early 1990's, with the growing popularity of the Internet, we have witnessed the birth of a new type of cybercrime; that is PHISHING. In 1987, a detailed description of phishing was introduced, and the first recorded attack was in 1995 (James, 2005). Deceiving users into giving their private information has a long tradition in the cybercrime community. At the beginning, phishers acted individually, or in small and simple groups. Surveys commonly depict

early phishers as mischief-makers aiming to collect information to make long-distance phone calls (Watson et al., 2005). In the early age of phishing, phishers mainly designed their attacks to deceive English-speaking users. Today, phishers have broadened their attack to cover users and businesses all over the globe (Sullins, 2006).

Usually, phishing is accomplished through the practice of social engineering. An attacker may introduce himself as a humble and respectable person claiming to be new at the job, a helpdesk person, or a researcher. An example of using social engineering is urgency by asking the user to submit his information as soon as possible. Risk of terrible results if the user denies complying is another tactic used to start social phishing. For instance, phishers warn the user that his account will be closed or the service will be terminated if he does not respond. However, some social engineering tactics promise big prizes by showing a message claiming that the user has won a big prize and to receive it he needs to submit his information. Nowadays, as monetary organizations have improved their online investments, the economic benefit of obtaining online account information has become much larger. Thus, phishing attacks have become more proficient, planned and efficient. Although phishers focus on individual customers, the organizations that phishers are mimicking are also victims because their brand and reputation is compromised.

Phishing is an alternate of the word *fishing*, and it refers to a bait used by phishers who are waiting for the victims to be bitten (James, 2005). One of the earliest methods of hacking was targeting telephone networks and it was called *Phone Phreaking*. This name was behind the origin of the *ph* replacement of the character *'f'* in the word *fishing*. There are several definitions of phishing; some definitions believe that phishing demands sociological skills in combination with technical skills, as in the definition that comes from the Anti-Phishing

Working Group (APWG) (Aaron & Manning, 2014 A): "*Phishing is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity and financial account credentials"*. Another definition comes from (Qi & Yang, 2006): "*A phishing website is a style of offence that network fishermen tempt victim with pseudo website to surrender important information voluntarily"*. A detailed description stated by (Kirda & Kruegel, 2005): "*Phishing is creating a fake online company to impersonate a legitimate organization; and asking for personal information from unwary consumers depending on social skills and website deceiving methods to trick victims into disclosure of their personal information which is usually used in an illegal transaction"*. Some definitions assume that the success of phishing websites depends on their ability to mimic a legitimate website, because most Internet users, even those having good expertise in the Internet and information security have a propensity to decide on a website's validity based on its look-and-feel, which might be orchestrated proficiently by phishers. An example of such definitions comes from (James, 2005): "*Phishing attempts to masquerade as a trustworthy entity in an electronic communication to trick recipients into divulging sensitive information such as bank account numbers, passwords, and credit card details"*. We may outline all the previous definitions in one sentence: "*A phishing website is the practice of creating a copy of a legitimate website and using social skills to fool a victim into submitting his personal information"*.

3.3. Phishing Techniques

Until recently, phishers have relied heavily on spoofed emails to start a phishing attack by urging the victims to reply with the desired information. These days', social networking websites are used to spread suspicious links to entice victims to visit phishing webpages. A recent study from Symantec Corporation (Nahorney, 2015) estimated that one phishing email occurs every

2,901.7 emails sent through their system daily in the finance sector. In another report (Symantec Corporation, 2013) it was shown that the number of phishing websites that imitate social networking websites' rose by 12% in 2013. If a phishers is able to obtain users' social networking login information, he can send out phishing emails to all their contacts. An email that seems to be initiated from a well-known person looks much more reliable. These days, novel phishing methods are becoming more frequent, such as Man-In-The-Middle attacks (MITM) (Keizer, 2007) and malware.

3.4. Life Cycle of Phishing Attacks

To tackle phishing, we have to thoroughly explore the nuts and bolts of the attacks. In this section, we will explain the life cycle of phishing attacks, which is shown in Figure 3.1.

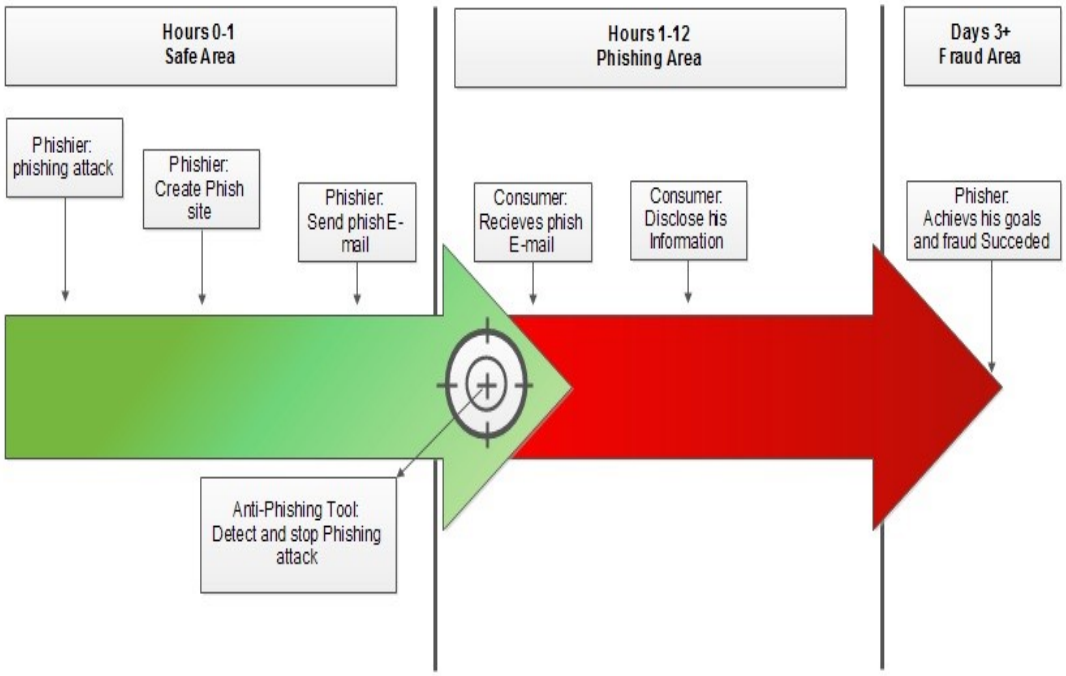


Figure 3.1 Phishing Websites Life Cycle

1. **Planning**: usually, a phisher starts planning for his attack by selecting his victims, the information to be collected and which method to use in the attack. The main aspects considered by phishers to pick their targets are how to obtain the maximum profits at the lowest costs and the least possible risk. Phishers might need to breach the organization calendar, a social networking website, or the employee list in an organization. A study (Jagatic et al., 2007) offered by ACM magazine showed that Internet users are 4.5 times more likely to be victims of phishing if they received an invitation to visit a URL link from a person they know. That explains why phishers target social networking sites. Email remains the main spreading channel for phishing URL links (Kaspersky Lab, 2013). However, phishing has spread beyond email to include Internet Relay Chat (IRC), forums, Instant Messaging (IM), vulnerable websites such as blogs, peer-to-peer file sharing, Voice over IP (VoIP), Short Messaging Service (SMS) and Social Networking Sites.
2. **Collecting**: the moment the user takes an action that makes him prone to credentials theft, he is then incited to submit his information through a trustworthy-looking web page. Commonly, the fake webpage is hosted on a compromised server, which has been exploited by the phisher for this purpose. Sometimes, phishers may use the free cloud applications, such as Google spreadsheets in order to host their fake webpages. No one will block spreadsheets.google.com or even google.com. Thus, not only naïve users will be conned, but also expert users' are less likely to block these websites.
3. **Fraud**: once the phisher have achieved his goal, he then becomes involved in fraud by impersonating the victim. Sometimes, the users' credentials are sold on the Internet black-markets. The amounts of activities that take place within the first few hours of a phishing life cycle are the most important

aspect of any attack. Once the fake website has been created and the phishing email has been sent, the anti-phishing tool should detect and stop the phishing website before the user submits his credentials as we recommend in Figure 3.1 above.

3.5. Significance of Phishing

A report disseminated by the APWG (Aaron & Manning, 2014 A) shows that the number of detected phishing websites reached a monthly high of more than 17,000 phishing attempts in December 2014. The total number of unique phishing reports submitted to APWG in the fourth quarter of 2014 was more than 197,000 URLs. This was an increase of 18% from the third quarter of the same year. The USA continued to be the top hosting country of phishing websites. The number of targeted brands increased to 300 brands in December 2014 after reaching 271 brands in October of the same quarter. The average number of phishing URLs per brand increased to 57.73 URLs in December 2014 after reaching 56.25 URLs in October. The ratio of IP-address based phishing URLs increased in this quarter to 2.4%. More than 20,000 unique phishing emails were sent monthly during that period. The most industrial sector targeted by phishers was the Retail/Service sector with 29.37%, followed by the Payment Services sector with 25.13%. The UK financial sector targeted heavily in this quarter, and high-profile targets such as Barclays, Halifax, and Santander were phished significantly in the final quarter of the 2014. However, Ihab Shraim, the president and Chief Technology Officer (CTO) at MarkMonitor[2] (MarkMonitor, 2013) say, "*it is unlikely that traditional phishing will stop since the cost of producing a phishing website is almost insignificant*". A survey disseminated by Gartner, Inc (McCall, 2007) reveals that phishing websites continue to escalate and cost US financial sector an estimated $3.2

[2] A service provider for the protection of corporate trademarks on the Internet.

billion annually. The same survey estimated that 3.6 million victims fall in such attack. In July 2013, a report distributed by The British House of Commons, Home Affairs Committee (Authority of the House of Commons, 2013) estimated that the overall cost of cybercrime to the UK is valued at £27 billion in 2012, with more than £600 million directly owing to phishing attacks. A poll of 2,000 US adults carried out by (Harris Poll, 2006) shows that 30% of those surveyed have limited their online transactions and 24% have limited their e-banking transactions because of phishing.

Several academic studies, commercial and non-commercial solutions are offered these days to mitigate phishing attacks. Some non-profit organizations such as APWG, PhishTank and MillerSmiles provide forums of opinions as well as distribution of some practices that can be organized against phishing. In May 2014, the free software download repository (Download.com, 2014) offered more than 400 anti-phishing tool. Yet, with the availability of this amount of precautionary measures, one might ask:

Why phishing still alive?

To answer this question we need to understand that a phishing attack is a combination of technical and social engineering practices. From the social engineering perspective, phishing is an example of a larger category of web threat known as semantic attack. Such attack focus on how a user interacts with computers or how he assigns meanings to emails and website contents (Liu et al., 2006). In this sense, the key principle in combating phishing websites is, *the Internet users have to inspect the security indicators within a website*. However, most Internet users lack the basic skills in recognizing such decisive indicators (Dhamija et al., 2006). In other words, users' tend to trust emails and webpages based on superficial trust information provided by phony clues within the email or the webpage. In general, phishers recognise that the Internet users' are the weakest link in the protection chain. However, from the technical perspective,

phishers know that the longer a phishing campaign uses the same set of features the more likely the anti-phishing parties will detect and deploy countermeasures against it. Phishers, therefore, adapt by constantly changing the set of features used to design such fake websites.

Overall, we believe that one possible solution to address this threat is by creating an intelligent model that reduces the human factor role. Such model needs to be updated regularly to keep pace with the latest phishing techniques.

3.6. Phishing and Concept Drift

The anti-phishing measures may take several forms including educational, technical and legal methods. In this study, technical methods are the subject of specific interest, precisely, intelligent heuristic-based approaches. The heuristic based approaches depend mainly on picking a set of decisive features obtained from the websites (Gastellier-Prevos et al., 2011). The method in which these features are processed is very important in producing a sound and effective decision. In many perspectives, such features are processed manually. By this means, after extracts some features, the user analyses the features and finally he decides on the website status. However, automating the classification process becoming an urgent need because tactics used become more complicated, phishing websites become more stylish, and the analysis time increases.

Data mining introduces itself as a promising technique that can produce sound and accurate decisions. Phishing websites problem can be posed as a supervised classification DM task that aims to learn a classifier that is able to assign each website to one of predefined classes according to some decisive criteria. Phishing can be thought as a binary classification problem since the target class has two possible values, i.e. phishing or legitimate. Yet, most of the

contemporary DM based anti-phishing models are devoted to a static learning strategy where the complete data sets are introduced to the DM algorithm (Abu-Nimeh et al., 2007), (Zhang et al., 2007), (Fette et al., 2007), (Miyamoto et al., 2008), (Basnet et al., 2008), (Aburrous et al., 2010 C), and (Abdelhamid et al., 2014). Such learning method is called offline learning. This learning method relies on assumptions of stationarity of the phishing websites problem, which must be strongly questioned (Ma et al., 2009) (Basnet et al., 2012). However, for domains where the training data set examples are constantly flowing as in phishing websites problem, the classification models should continuously be updated because offline classification models of a pre-determined structure might be prone to concept drift (Polikar et al., 2001). The concept drift that occurs in phishing websites classification problem is considered a cyclical concept drift in the sense that the set of features that are used in predicting phishing might disappear at specific time and return to reappear again in the future.

In (Basnet et al., 2011), the authors suggest identifying a significant and representative subset of phishing websites features as soon as a new data set becomes available to create a new classification model. Yet, although the experimental results look promising, this solution might result in the so-called catastrophic forgetting. Another solution (Ma et al., 2009), suggests merging the newly collected data set with all historical ones and the resulting data set is used to create a new model. Nevertheless, this solution might be a high memory cost solution.

In general, phishing websites is a dynamic supervised classification problem that can be characterized as a constantly evolving phenomenon. Therefore, an effective anti-phishing model is needed and must be improved regularly in order to cope with the evolving nature of the problem.

3.7. Examples of the Virtual Concept Drift that Occurs in Phishing Websites

To ensure long life, any classification model should be updated if there is a catalyst to do so. In phishing websites problem, the constant changing in the input features is considered the catalyst for modifying the model. As long as the input features are constantly changing, the classification model should also be changed.

One example of the catalysts for modifying the anti-phishing classification model emanates from the utilization of the redirect URL feature. In their article (Aburrous et al., 2010 A) the authors reveal that redirect URL is ineffective in predicting phishing websites since it has appeared 10% only in their data sets. Nevertheless, these days a new phishing method that is based mainly on a delayed redirect URL has emerged. Such technique is called Tabnabbing (Mesh, 2013). If the user visits the doubtful webpage, a message will appear asking him to wait while the webpage is loaded. Meanwhile, the user may switch to another tab because he is a multi-tasking. As soon as the malicious webpage realize that another tab is opened it will redirect itself to a phishing webpage. When the user goes back to the tab where the malicious webpage is loaded, he might think that he signed out of his email or his business application, thus, he will write down his credentials in the fake webpage. Once the credentials are entered, they are grabbed by the phisher and the victim is redirected back to the legitimate webpage. Another example comes across the use of long and tiny URL features. Some techniques assume that long URL is an important clue about the website legitimacy (McGrath & Gupta, 2008), (Aburrous et al., 2010 B), (Basnet et al., 2011). However, phishers can make phishing URLs less suspicious by using tiny URL method. Hence, long URL might become insignificant in revealing phishing websites. The use of iframe feature is another example of the catalysts for modifying the anti-phishing

model. In their article (Alkhozae & Batarfi, 2011) the authors confirmed that iframe is a significant feature in predicting phishing websites. Yet, iframe might also be used to deliver advertisements to users.

However, the virtual concept drift occurring in phishing websites might happen deliberately. If phishers were able to recognize the set of features employed in predicting phishing websites within a particular anti-phishing tool, then their mission in designing a new phishing website could become easier. For instance, they can modify the way in which a specific feature is normally used, or invent new features that can circumvent the anti-phishing tool. This rationale explains why phishers directly target anti-phishing organizations. More specifically, in 2009, the hackers who breached Google's network were able to steal the source code for the company's global password system (Zette, 2010). Although the users are not directly affected, the hackers could study the software for security vulnerabilities to devise methods to breach the system, which may have consequential ramifications on users, as highlighted in the New York Times (Markoff, 2010). In this digital era of tremendous globalization, hackers phish for the source code of leading technology companies such as Microsoft, Apple, Oracle and Adobe, in order to use in future sophisticated and targeted attacks, as mentioned by FireEye's Chief Executive Officer, David Dewalt (Stevenson, 2014). According to Dewalt, "Phishers go for source code as if they can get the source code and find a hole to get round users' defenses".

3.8. Abilities of Offline Intelligent Techniques in Predicting Phishing Websites

Conventional offline DM techniques are widely applied to become more viable in predicting phishing. Some studies evaluate the performance of several DM and ML algorithms in detecting phishing, such as (Fette et al., 2007), (Abu-

Nimeh et al., 2007), (Basnet et al., 2008), (Miyamoto et al., 2008) and (Mohammad et al., 2013-A). Several DM and ML algorithms, feature sets, and evaluation criteria are utilised in these studies. The results indicate that although offline DM and ML prove their merits, there is no optimal algorithm outperforming all others in all cases.

Most importantly, phishers know that the longer a phishing campaign uses the same set of features the more likely the anti-phishing parties will detect and deploy countermeasures against it. Phishers, therefore, adapt by constantly changing the set of features used to design such fake websites. Therefore, although offline classification algorithms have produced good prediction results in many contexts, they have been inefficient for long life because phishing techniques constantly changing (Ma et al., 2009) (Basnet et al., 2012).

Moreover, these methods do not take into consideration the possibility to accommodate new knowledge whenever new training data sets are acquired since the learning phase is a one-time process. Therefore, the performance of the conventional offline DM and ML based anti-phishing models may fall remarkably if some domain attributes change. Most importantly, the learning process in such techniques assumes that the data sets are completely offered before the training phase takes place; however, that is not always true, since training data set is often obtained over time in streams of single instances or a batch of instances. Such scenario raises an important dilemma, since the set of features that best predicts the class variables may differ from one batch to another, hence, a virtual concept drift occurs (Ma et al., 2009) (Basnet et al., 2012).

However, it is difficult to anticipate what these new features will be or when they might appear. As a result, it would be difficult to determine a specific size

of the training data set that should be collected in order to create a new classifier.

In general, the one shot *store-now* and *mine-later* techniques are often impractical in domains where the underlying data are constantly changing (Hoens et al., 2012) as in phishing websites problem (Ma et al., 2009) (Basnet et al., 2012). Learning from such data sets demands a model that can be updated continuously in order to derive benefits from the newly available data. Such model should also maintain its performance on the old data, i.e. it does not forget the previously learnt knowledge.

3.9. Anti-Phishing Approaches

Identifying phishing websites is an essential step towards protecting Internet users against posting their sensitive information online. Several approaches have been suggested to tackle phishing. Anti-phishing methodologies can be grouped into seven categories: human based protection approach, legal solutions, blacklist and whitelist based approach, instantaneous based approach, decision supporting tools, community rating based approach, and intelligent heuristic based approaches. Below, we shed the light on common anti-phishing techniques by evaluating a list of related works and substantiating the need for an automated technique, as opposed to human involvement when fighting against phishing.

3.9.1. Human based Protection Approach

One strategy for fighting phishing attacks is by educating Internet users to recognize phishing attempts rather than just warning them about possible risks. In (Dhamija et al., 2006) a usability study conducted to understand how and why phishing works shows that lack of attention to some security indicators, the lack of knowledge of computer systems, and lack of attention to

the absence of some security indicators are the main reasons of why users become victims of phishing. In (Ronald et al., 2007), the authors reveal that training and educating Internet users about web security is one of the most essential aspects of an organization's security situation. A study conducted at Indiana University (Jagatic et al., 2007) shows that users are more likely to fall victims of deception when they are emailed by someone they knew. In 2007, a survey was conducted to measure the users understanding of the malicious contents within the webpages (Julie S. et al., 2007). The survey found that despite that several participants evinced some technical background they refused to enter their personal information into legitimate webpages due to the potential of severe results. An interesting education technique that essentially uses the game based learning practice was proposed in (Sheng et al., 2007). This technique offers a close link between action and instantaneous feedback. The authors have created an online game called *Anti-Phishing Phil*. The study shows that a user who plays this game becomes more cautious of phishing websites. An email based education technique called *PhishGuru* (Kumaraguru et al., 2007) that educates Internet users how to use evidences in URLs to evade phishing attacks suggests that alongside the automated detection systems users education may offer a complementary method to help Internet users to identify deceptive emails and webpages. A recent study (Sheng et al., 2010) founds that the users might rely on superficial heuristics in determining how to reply to emails. For instance, some users assume that since the organisation they are dealing with already has their information it would be safe to give it again. The same study shows that age and gender are two key demographics to predict phishing.

Overall, although education might be a good method in fighting phishing it requires high costs. Not all organizations are able to pay out extra money on users' education, given that users' education is not a one-time cost. In addition,

it is not sure that after appropriate education the users' will act in an ideal way. Moreover, this approach requires a long-winded process and users' have to dedicate a substantial amount of their time to studying the phenomenon. Further, phishers are becoming more skilled in mimicking genuine webpages even to the extent that security experts might be deceived.

3.9.2. Legal Solutions

Followed by many countries, the USA was the first to enact laws against phishing activities. Phishing has been added to the computer crime list for the first time in January 2004 by the Federal Trade Commission (FTC)[3]. In May 2006, the US president George W. Bush gave his orders for the establishing of the *President's Identity Theft Task Force* (Executive-Order-13402, 2006) that aims to ensure that the efforts of the Federal Authorities become more effective and more efficient in the area of preventing and identifying cybercrime attempts. In January 2005, the General Assembly of Virginia added phishing to its computer crimes act (General Assembly of Virginia, 2005). In March 2005, the anti-phishing act was introduced in the US Congress by Senator Patrick Leahy (Gross, 2004). In 2006, the UK government strengthened its legal arsenal against fraud by prohibiting the development of phishing websites and enacted penalties of up to 10-year imprisonment. In 2005, the Australian government signed a partnership with Microsoft to teach law enforcement officials how to combat different cybercrimes.

Although law enforcement officials have successfully arrested, prosecuted and convicted some phishers for the past few years (BBC News, 2005) (TrendMicro, 2013), a criminal act does a poor job of preventing phishing since it is hard to trace phishers. In addition, phishing attacks can be performed quickly and later

[3] A US government agency that aims to promote consumer protection.

the phisher may disappear into cyberspace. Therefore, law enforcement authorities must act rapidly because on average most phishing websites live only for 54 hours (Dhamija et al., 2006).

3.9.3. Blacklists and Whitelists based Approach

A blacklist is a list of URLs thought to be malicious. Several methods might be used to set up a blacklist, such as heuristics from web manual voting, crawlers, and honeypots[4]. Whenever a website is visited, the browser refers it to the blacklist to examine if that website exists within the list. If so, the website is a malicious website, and hence, the browser warns users of a possible attack. Blacklists can be saved either locally on the user's machine or on a server that is queried by the browser for every requested URL. The main aspects of blacklists are quantity, quality and timing. Quantity refers to the amount of phishing URLs that are available within the list. On the other hand, quality can be measured in terms of erroneous listing and is commonly known as the false positive rate. The third and most significant aspect is timing. If the blacklist updating process is slow, this will give the phishers the opportunity to carry out their attacks. Blacklists are updated at various speeds. In a recent study (Sheng et al., 2009) the authors estimated that approximately 47% - 83% of phishing URLs are displayed on blacklists almost 12 hours after they are launched. The same study ascertained that zero hours defence delivered from most well-known blacklist based toolbars claimed a true positive rate ranges from 15% to 40%. A survey published by APWG (Rasmussen & Aaron, 2010) revealed that 78% of phishing domains were hacked domains and at the same time, they were already serving legitimate websites. Consequently, blacklisting those domains will in-turn add legitimate websites to the blacklist. Even if phishing webpages are removed from the blacklisted domain, legitimate

[4] A trap set to detect and counteract attempts at unauthorized use of information systems.

webpages hosted on the same domain might be left on the blacklist for a long time, thus causing significant harm to the reputation of the legitimate website or organization. Several solutions that are using blacklists approach have been deployed these days, one of which is Google Safe Browsing (Google-Safe-Browsing, 2010). Another solution is Microsoft Internet Explorer 9 anti-phishing protection (Microsoft-Support, 2012). Site Advisor (McAfee SiteAdvisor, 2006) is a database-backed measure that is designed essentially to defend against malware based threats such as Trojan horses and Spyware. Site Advisor comprises automated crawlers that browse websites, then carry out tests and build threat assessments for every visited website. Regrettably, like other blacklists, Site Advisors cannot recognize newly created threats. VeriSign (Symantic, 1995) has a web-crawler that collects millions of websites to recognize clones to discover phishing webpages. One potential drawback with crawling and blacklist approaches might be that anti-phishing parties will always race against attackers. Netcraft (Netcraft Toolbar, 1995) is a small software package that is activated every time a user browses the Internet. Netcraft relies on a blacklist which consists of fraudulent websites recognized by Netcraft and those URLs submitted by the users and verified by Netcraft. The main problem with Netcraft is that the final decisions regarding website legitimacy are primarily made by the Netcraft server, rather than the user's machine. Accordingly, if the connection to the server is lost for any reason, then the user will be under threat during this period.

The study in (Ludl et al., 2007) analyses and measures the effectiveness of two popular anti-phishing solutions based on blacklists, namely:

1. Blacklists preserved by Google and used by Mozilla Firefox.
2. Blacklists preserved by Microsoft and used by Internet Explorer.

The data set used to conduct the experiments consisted of 10,000 phishing URLs. The first experiment was aimed to study the effectiveness of both blacklists. Only Google gave back proper responses for all URLs in the data set. On the other hand, Microsoft server accurately responded to only 60% of the data set. One more experiment was conducted taking into consideration those URLs for which both servers returned a response and which were online at the time of conducting the experiment. The results showed that Google was capable of labelling 90% of the data set items correctly, whereas Microsoft only labelled around 67%. An additional experiment was conducted to assess the time needed to update the blacklist for both servers by testing URLs that were not initially blacklisted but added after some period of time to the list. The results showed that Microsoft blacklist was updated after 9 hours and 7 minutes at the best; however, at worst, it was updated after 9 days and 6 hours. Google's fastest update took about 20 hours and the slowest update time was almost after 12 days.

The authors in (Sharifi & Siadati, 2008) suggested a method to shape blacklists automatically by making use of search engines such as Google search engine. The proposed method starts by extracting the company name from the suspected URL, and then the search engine is used to search for the extracted company name. If the suspected URL is shown in the first 10 returned results, then the URL is considered a legitimate URL, otherwise, it is considered a phishing one; hence, it will be added to the blacklist. However, the same study acknowledges that this method introduces additional delay in the Internet browsing.

Whitelists are the opposite term to blacklists. A whitelist consists of a set of trusted URLs, whereas all others are thought undependable. Whitelisting is an approach that encompasses a new identity problem since a newly visited

website is initially marked as malicious. However, to overcome this problem, all websites expected to be visited by the user must be listed within the whitelist. However, it is virtually impossible to anticipate where a user might browse. A solution could be achieved by using dynamic whitelisting whereby a user is involved in creating the list independently. For every visited website, the user must decide whether to add this website to the whitelist or not. However, if phishing websites persuade users to submit their sensitive information, they could also positively persuade them to add it to the whitelist.

The authors in (Chen & Guo, 2006) proposed an Automated Individual Whitelist (AIWL), which is an anti-phishing tool based on an individual user's whitelist of known trusted websites. AIWL traces every login attempt by individual users. In case a repeated successful login for a specific website is achieved, AIWL prompts the user to add the website to the whitelist. However, this technique assumes that users only submit their credentials to legitimate sites, whereas all others are considered malicious.

Another solution that primarily depends on the whitelist technique was presented in PhishZoo (Afroz & Greenstadt, 2011). This technique builds profiles of trusted websites based on fuzzy hash techniques. A profile of a website is a fusion of several metrics that exclusively identify a specific website such as URL, images, HTML code, scripts and SSL certificate. As soon as a new website is visited, PhishZoo builds a profile for that website. The new profile is compared to the profiles in PhishZoo database. If PhishZoo finds an identical copy of the loaded website in PhishZoo database, then this website is considered legitimate. Otherwise, if PhishZoo finds a partially matches, then:

1. If a website's profile does not match, but SSL certificate and addresses do, then PhishZoo will update the profile stored on the PhishZoo database.

2. If website's profile match, but SSL certificate or addresses do not match, then PhishZoo will assign phishing to the website.

However, if PhishZoo does not find any matching, then it will prompt the user to build a new profile. Yet, PhishZoo revealed that most phishing websites were simply copies of real websites. Nonetheless, if the loaded website, which could be a phishing website, does not look like its imitation (either by changing the size or position of the website logo) then PhishZoo will ask the user to decide on the website's legitimacy.

3.9.4. Instantaneous based Protection Approach

Online transaction systems consist of three components: the users, the websites (from which all transactions are performed) and the stock database. We believe that the Instantaneous based Protection Approach (IBPA) protects the migration of sensitive information during the transaction process by protecting the source or the destination of the information or sometimes protecting both the source and destination. Users are considered the source of information, whereas websites are considered the destination. IBPA protects sources of information by either authenticating user's credentials or protecting the input information instantly. However, to protect the destination of information, the IBPA authenticates the website or server so that the user is assured that the website he is dealing with directly corresponds with the website where his credential will migrate.

A white paper published by Cryptomathic (Cryptomathic, 2012) categorizes user authentication mechanisms into three types, as follows:

1. Something the user knows such as a password, a secret code or a PIN number.
2. Something physical such as a fingerprint or an iris scan.

3. Something a user owns, for instance a credit card or a token generator.

Phishing arises from a reliance on the first category. A strong authentication may be achieved using two completely diverse credentials proof. This is often referred to as Two-Factor-Authentication (2FA). Typically, 2FA generates and displays a one-time password (OTP) which is valid for one time use only or sometimes for a specific period of time to ensure that the user not just fills in his password but also he uses an OTP. However, 2FA is a server-side approach; hence, if the server breaks down, then the user would not be able to access his account. Moreover, although 2FA decreases the risk of phishing attempts, phishers have invented some circumventing techniques such as switching to real-time MITM attacks using malware techniques. America Online (AOL) distributes RSA[5] SecurID devices (RSA, 2012) as per Figure 3.2 to AOL users. This device produces a unique 6-digit token every minute. The user should submit his password along with this token in order to log into his account.

Figure 3.2 RSA SecurID Device

Hardware token based technique is considered quite costly since each user should have his own device. In their article, Mannan and Oorschot recommend using mobile phones to verify identity on the Internet (Mannan & Oorschot, 2007). Similarly, (Mizuno et al., 2005) proposed multiple communication channels to authenticate user's identity. Their solution enables Internet Service Providers (ISPs) to use trusted communication channels to verify the user's

[5] RSA stands for Rivest, Shamir, and Adelman, the inventors of the technique.

identity on non-trusted modes of communication. The authors also suggested using mobile phones as an example of trusted communication channels.

Most Internet users use one single password for several personal accounts. Therefore, if phishers are able to break low security websites, then they can obtain thousands of username and password pairs and use them on secure e-commerce websites such as Amazon, eBay and PayPal. In (Ross et al., 2005) the authors suggested a solution for this problem by creating a browser extension called PwdHash which instantly alters a user's password into domains specific password that consists of the pair (Password, Domain name). Hence, the user can securely re-use the same password on several websites. However, the safest way to shield users against phishing attacks is to predict phishing websites, rather than encrypt user password.

A browser sidebar used for handling user logins was proposed by (Wu et al., 2006) as per Figure 3.3**.** The users are advised to submit any sensitive information using the Web Wallet sidebar and not through website forms. The Web Wallet has some similarities to Microsoft InfoCard identity meta-system (Brown, 2006) since it requires users to enter their information via an authentication interface. However, there are several variations, as websites should be modified to accept InfoCard, whereas Web Wallet is used with web browsers. InfoCard requires support from identity providers, i.e. banks, credit card issuer and government agencies. InfoCard users are compelled to get InfoCard from various identity providers and users should authenticate themselves whenever they choose InfoCard.

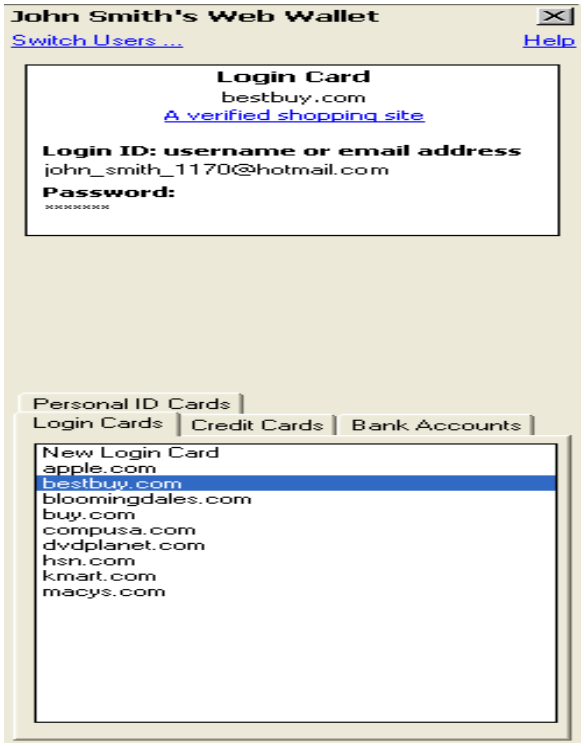


Figure 3.3 Web Wallet SideBar

Several techniques protect user's credentials by adding some random information to the original credentials so that it becomes quite hard for phishers to isolate real credentials. These approaches decide whether a website is valid or not by examining the consistent HTTP response code which could be either 200 Success or 401 Authentication Failed. A website is classified as phishing, if the response is always success or failure on all retries.

A new scheme, called Dynamic Security Skins (Dhamija & Tygar, 2005), allows a remote server to prove its identity to the user by showing a secret image. The strength of this schema comes from the difficulty for the phishers to spoof. This method requires the user to make verification based on what image he expects with an image generated by the server. One drawback of this schema is that the user bears the burden of deciding whether the website is phishing or not. Moreover, this approach suggests a fundamental change for both servers and

clients within the web's infrastructure, which therefore can only succeed if the entire industry supports it. In addition, this method does not offer security if the user accesses his account from a public workstation. The advantage with this approach is that the user does not need to understand what the digital certificates are; or any technical aspects of the security mechanisms. The user simply has to verify that the generated image and the received one are identical.

Kirda and Kruegel proposed an approach that instantly monitors the data flow, such as passwords and credit card information (Kirda & Kruegel, 2005). The authors create a tool referred to as *AntiPhish* which is a Mozilla Firefox extension. This approach assumes that each domain is related to only one password on each specific machine. Thus, if the user visits a website and types in his password, a list of random domain and password pairs is created. Then, AntiPhish keeps watching the password fields on any visited websites, and searches the domain of that website among a list of previously visited ones, when an identical password is found, AntiPhish warns users of potential attacks since the same password is entered on two different pages. The main drawback of this tool is that the false positive rate may increase if identical passwords are used on multiple websites, which what users usually tend to do. In addition, if the user has more than one account on the same website an unwanted warning message will appear.

3.9.5. Decision Supporting Tools

These tools do not make any decision on the website legitimacy, but they extract some features from the websites and clarify them to the user. However, the final decision on the website legitimacy is made by the user, i.e. a human factor. The user's experience and attention to the clues displayed by decision supporting tools are the decisive factors in making the final decision. An

example of decision supporting tools is SpoofStick (Spoofstick, 2005) which is a toolbar that can be installed on both Mozilla Firefox and Internet Explorer. SpoofStick shows the website's actual domain name. For instance, if the user visits a URL such as *www.ebay.com.spoofone.ca* then SpoofStick will consider *spoofone.ca* to be the domain of that URL. SpoofStick performs reverse DNS query to display the real IP address of the website. A user can take advantage of this information to decide whether the website is a legitimate website or not.

(Herzberg & Gbara, 2004) developed a Mozilla add-on called Trustbar as shown in Figure 3.4. Trustbar shows some information about the website credentials such as website name, logo, owner, certifying authority, or a warning message for unprotected websites. One of the drawbacks of Trustbar is that it shows the website's Certification Authority (CA) without checking its trustworthiness, and the user has to do such verification. Unfortunately, most Internet users have no idea about how to do such verification.

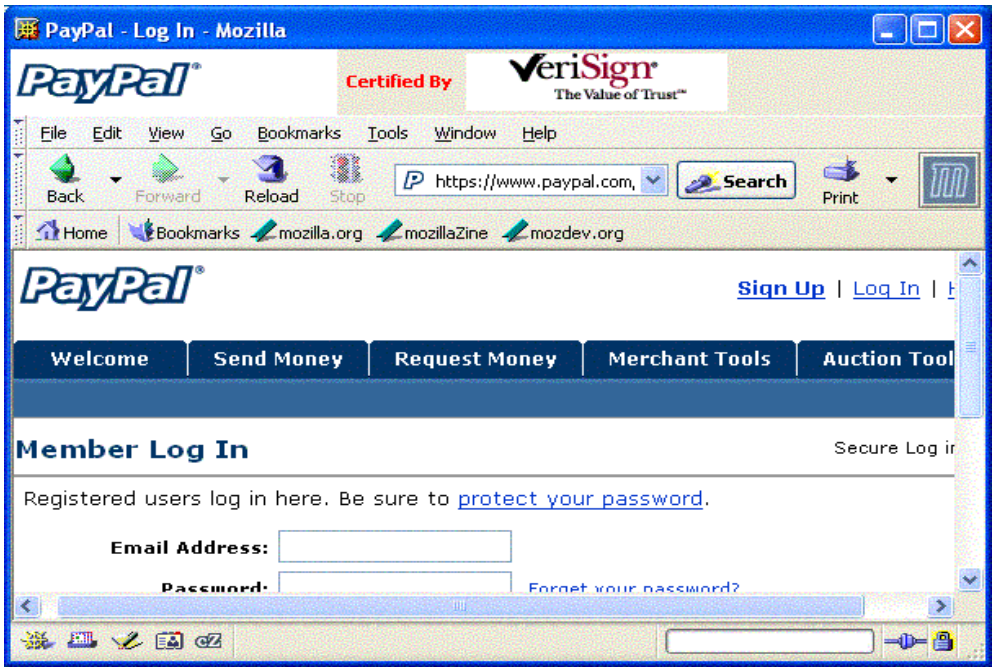


Figure 3.4 TrustBar

iTrustPage (Ronda et al., 2008) performs a Google search using the webpage's key-terms. However, iTrustPage does not solely rely on extracting heuristics, then judging on the website's legitimacy, but it partially depends on the user's input to make the right decision because the search terms are provided by the user. When the user encounters a website that does not exist in the iTrustPage's predefined whitelist, he is prompted to a collection of search terms that may be used to make a Google search. In general, Decision-supporting tools eventually depend on the user's skills to take the right decision on the website's legitimacy.

3.9.6. Community Rating based Approach

This approach relies on the user's experience to decide whether a website is phishing or not. Organizations such as APWG (APWG, 2003), Federal Trade Commission (FTC) (FTC, 2015), and PhishTank (PhishTank, 2011), have introduced cooperative enforcement programs to fight phishing and identity theft. PhishTank (PhishTank, 2011) was launched in October 2006. The main goal of PhishTank is to provide the parent company OpenDNS (OpenDNS, 2006) with a reliable phishing data set. PhishTank makes all its databases open and accessible. Thus, developers and organizations such as Yahoo, Mozilla, and Microsoft as well as leading academic institutions, can make use of such databases. Cloudmark is another example of community based anti-phishing approach (Cloudmark Inc, 2002). The fundamental principle in Cloudmark is "*if you visit a website which you recognize to be unsafe, just click the block button to warn other users*". Whenever a user is browsing the Internet, he can rate the website as *Good* or *Unsafe*. The users themselves are also evaluated based on their history of accurately labelling phishing websites. The user's reputation increases if he accurately blocks a fraudulent website and unblocks a legitimate one. Web of Trust (WOT, 2006) is a community based safe surfing tool that uses

an intuitive rating system to keep users safe while they shop, browse, and search online. For every searching attempt, WOT provides ratings to the searching results. The rates are updated constantly by various members of the WOT community and from various reliable resources, such as PhishTank (PhishTank, 2011), hpHosts (Malwarebytes, 2005), Panda Security (PandaSecurity, 1990), LegitScript (LegitScript, 2007) and TRUSTe (TRUSTe, 1997).

Overall, the effectiveness of the community based rating approach depends on the honest and active work done by users. In addition, some legitimate websites may not be rated, and thus, they may be considered bad websites. Moreover, some websites might be good, and then they might become bad and vice versa.

3.9.7. Intelligent Heuristics based Approach

This approach collects a set of features that can separate phishing websites from legitimate ones, then an intelligent model is trained using the collected features, and finally the trained model is used to recognize phishing websites in the real world. Selecting the features set is an essential step towards achieving a model with high accuracy. Several features have been suggested for detecting phishing websites; nevertheless, some of these features do not seem to be sufficiently discernment. Most importantly, phishers keep changing the features space constantly. Hence, a good intelligent anti-phishing model should constantly be updated.

The study in (Aburrous et al., 2010 B) is based on experimentally comparing associative classification algorithms. The authors have gathered 27 different features from various websites as shown in Table 3.1.

Table 3.1 Features used in Fuzzy based model

Category	Phishing factor indicator
URL & domain identity	Using IP Address
	Request URL
	URL of anchor
	DNS record
	Abnormal URL
Security & encryption	SSL certificate
	Certification authority
	Abnormal cookie
	Distinguished names certificate (DN)
Source code & Java script	Redirect pages
	Straddling attack
	Pharming attack
	Using onMouseOver
	Server form handler
Page style & contents	Spelling errors
	Copying website
	"Submit" button
	Using pop-ups windows
	Disabling right-click
Web address bar	Long URL address
	Replacing similar characters for URL
	Adding prefix or suffix
	Using the @ symbol to confuse
	Using hexadecimal character codes
Social human factor	Much emphasis on security and response
	Generic salutation
	Buying time to access accounts

These features range among three fuzzy set values (Legitimate, Genuine and Doubtful). The experimental results demonstrated that Domain Identity features and URL based features are the most significant features in predicting phishing websites. However, the Page Style has a minor impact on Social Human Factor criteria. Later, the authors use their 27 features to build a model to predict websites class based on fuzzy DM (Aburrous et al., 2010 C). Although their method is a promising solution, the authors do not clarify how the features are extracted from the website and specifically the features related to human factors, i.e. Much Emphasis on Security and Response, Generic Salutation, and Buying Time to Access Accounts. Moreover, the authors classify the website as very-legitimate, legitimate, suspicious, phishy or very-

phishy, but they do not explain the fine line that separates one class from another. In general, fuzzy DM uses approximations, which are not considered a good candidate for managing domains that require extreme precision (Sodiya et al., 2007).

An innovative model proposed by (Pan & Ding, 2006) is essentially based on capturing abnormal behaviours demonstrated by phishing websites. The model consists of two components:

1. Identity Extractor: which is an abbreviation of the organization's full name and/or a unique string appears in its domain name.
2. Page Classifier: this utilises some structural features that cannot be freely fabricated, such as the features relevant to website identity.

In their experiments, the authors have selected six structural features, i.e. Abnormal URL, Abnormal DNS record, Abnormal anchors, Server form handler, Abnormal cookies and Abnormal certificate in SSL. Support Vector Machine classifier was employed to decide on the website legitimacy. The experiments showed that the Identity Extractor presents better results when it comes across phishing websites because the legitimate websites are independent, whereas most of the phishing websites are interrelated (Pan & Ding, 2006). Moreover, the performance of the Page Classifier depends mainly on the results extracted from Identity Extractor. The overall classification accuracy of this method was 84% which is relatively considered low. However, this method snubs some features that can play a key role in determining the legitimacy of the website, which explains the low detection rate. One solution to improve this method could be by using additional features.

The method proposed in (Zhang et al., 2007) suggests utilizing CANTINA technique, which is content based technique to detect phishing websites using

the Term Frequency Inverse Document Frequency (TF-IDF) measures (Manning et al., 2008). CANTINA stands for **Ca**rnegie Mellon **Anti**-phishing and **N**etwork **A**nalysis Tool and it examines the webpage contents, then decides whether it is phishing or not by using TF-IDF. TF-IDF produces weights that assess the word importance to a document by counting its frequency. CANTINA works as follows:

1. Calculate the TF-IDF for a given webpage.
2. Take the five highest TF-IDF terms and add them to the URL to find the lexical signature.
3. Feed the lexical signature into a search engine.

If the *N* tops searching result contains the current website, hence, the website is a legitimate website. If not, on the other hand, it is a phishing website. *N* was set to 30 in the experiments. However, if the search engine returns zero results, then the website is labelled as phishing. This argument is the main drawback of using such technique as stated by the authors. To overcome this weakness, the authors combined TF-IDF with some other features, which are: Age of Domain, Known Images, Suspicious URL, Suspicious Link, IP Address, Dotes in URL and Using Forms. However, most legitimate websites contain images; thus, using the TF-IDF may not be right. In addition, this approach does not deal with the hidden text, which might be effective in detecting the type of the webpage.

Another method that utilises CANTINA with additional attributes is proposed in (Sanglerdsinlapachai & Rungsawang, 2010). The authors have used 100 phishing websites and 100 legitimate ones, which are considered limited in their experiments. The same set of features as in CANTINA has been used in this study, but with some changes as follows:

1. The Using-Forms feature has been considered as a filter to decide whether to start the classification process or not.
2. The Known Image and Domain Age features have been ignored.
3. A new feature has been suggested that is Domain Top-page Similarity.

The authors have performed three experiments. The first experiment evaluates a reduced CANTINA feature set. The second experiment tests whether the newly proposed feature is significant in detecting phishing websites. The third experiment evaluates the prediction accuracy after adding the new feature to the reduced CANTINA features. The results show that the new feature significantly improves the detection rate. The most accurate algorithm was NN, followed by SVM and Random Forest whereas Naïve Bayes gave the worst result.

Typically, toolbar based approach relies on blacklists to judge on the website validity. However, some toolbars decide on the website legitimacy by extracting a set of features from the website, then make some calculations and finally produce the final decision on the website status. One example of such toolbars is shown in Figure 3.5 and is called SpoofGuard (Chou et al., 2004). SpoofGuard calculates a spoof index and alerts the user if the index goes beyond a predefined threshold.

Figure 3.5 SpoofGuard Toolbar

SpoofGuard uses configurable weighted heuristics to determine the likelihood that a website is a malicious website. The user is able to configure the weights for each feature as shown in Figure 3.6. One negative aspect of SpoofGuard is

that if the user is warned of a possible phishing attack, he could override this warning, hence, no more warnings will appear in the future for any webpage having the same domain name. However, a phisher may create a harmless website and later the truthful website is replaced with a malicious one; thus, SpoofGuard will not warn the user of possible attacks. This kind of attack is called Bait and Switch Attack.

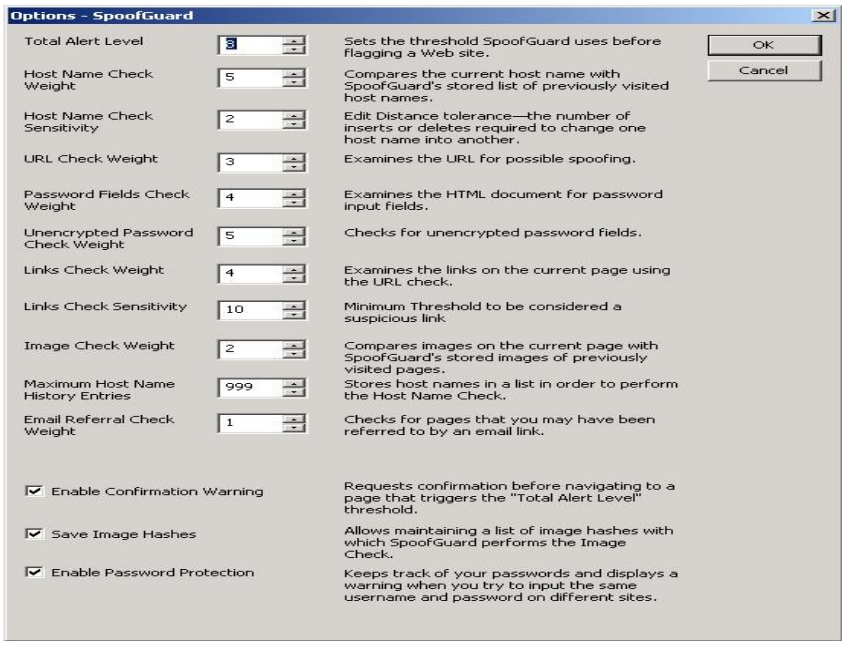


Figure 3.6 Configure Features Weights Window

Some organizations offer toolbars to their customers to keep them safe from being phished. An example of such toolbars is the eBay toolbar (eBay Toolbar's, 1995). This toolbar is solely designed to protect eBay and PayPal users only. However, like SpoofGuard, this toolbar is prone to a Bait and Switch attack.

In 2006, a study aimed to evaluate the effectiveness of five toolbars (Wu et al., 2006), those are: SpoofStick (Spoofstick, 2005), Netcraft (Netcraft Toolbar, 1995), Trustbar (Herzberg & Gbara, 2004), eBay Account Guard (eBay Toolbar's, 1995) and SpoofGuard (Chou et al., 2004). The results reveal that there are two reasons why the users' fall victims to phishing: Firstly, most Internet users

discard what the toolbars display because they tend to judge on the website class based on its look-and-feel. Secondly, several companies do not follow sensible practice in designing their websites. Hence, the toolbars cannot help the Internet users' to differentiate between a poorly designed website and a malicious one.

URL and HTML DOM objects are not the only places to extract the phishing website features, but also the features might be extracted from visual related elements. The basic idea is that detecting phishing websites is similar to plagiarism and duplicate-document detection, except that phishing detection focuses on visual similarities whereas plagiarism detection focuses on text based similarity. One promising method proposed by (Wenyin et al., 2005) suggests detecting phishing websites based on visual similarities between phishing and legitimate websites. Initially, this technique decomposes the webpage into salient block regions depending on visual cues. The visual similarity between phishing and legitimate webpages is then evaluated in three metrics: block level similarity, layout similarity, and overall style similarity. A webpage is considered a phishing attempt if any metric has a value higher than a predefined threshold. However, this technique might be inaccurate because of the high plasticity of the webpage layout.

The authors in (Liu et al., 2006) suggested a phishing detection model using the Earth Mover's Distance (EMD) (Rubner et al., 1998). EMD is a measurement technique to assess the distance between two probability distributions. This approach assesses the similarity between phishing and legitimate websites at the pixel level of the websites without looking at the source code similarities.

3.10. Chapter Summary

In this chapter, we have shed the light on what phishing websites mean, the phishing websites life cycle, and we introduced some statistics that show the effect of phishing websites on the Internet community. In addition, several anti-phishing methodologies have been discussed. Such methodologies are categorized into seven groups. In addition, in this chapter, we showed that most intelligent anti-phishing techniques are based on an offline learning schema, according to which once the anti-phishing tool is produced it can obtain no further knowledge. Concept drift and catastrophic forgetting dilemmas have been discussed as the main issues to be addressed when designing an anti-phishing model.

CHAPTER 4

A Classification Framework based on Ensemble Self-Structuring Neural Network

4.1. Introduction

A challenging problem when designing a classification model for dynamic domains (as in phishing websites classification problem) is how to make the model learns continuously from evolving data sets. Resolving this issue demands a technique that can trace any changes that might affect the classification model overall accuracy; hence, revising the decision boundaries accordingly. Such a situation requires a learning schema that offers balance between stability and plasticity.

In this chapter, a new classification framework that continuously obtains knowledge from evolving data sets is proposed and is called Ensemble Self-Structuring Neural Network (ESSNN). The framework shown in Figure 4.1 suggests that as soon as a new data set batch arrives at time t and after confirming the presence of a virtual concept drift, a new classifier c_t is created. The newly derived classifier is combined with all previously created classifiers to produce a band of classifiers (ensemble of classifiers) C_t. The decision (assigning class value to test data) of C_t is a collective decision.

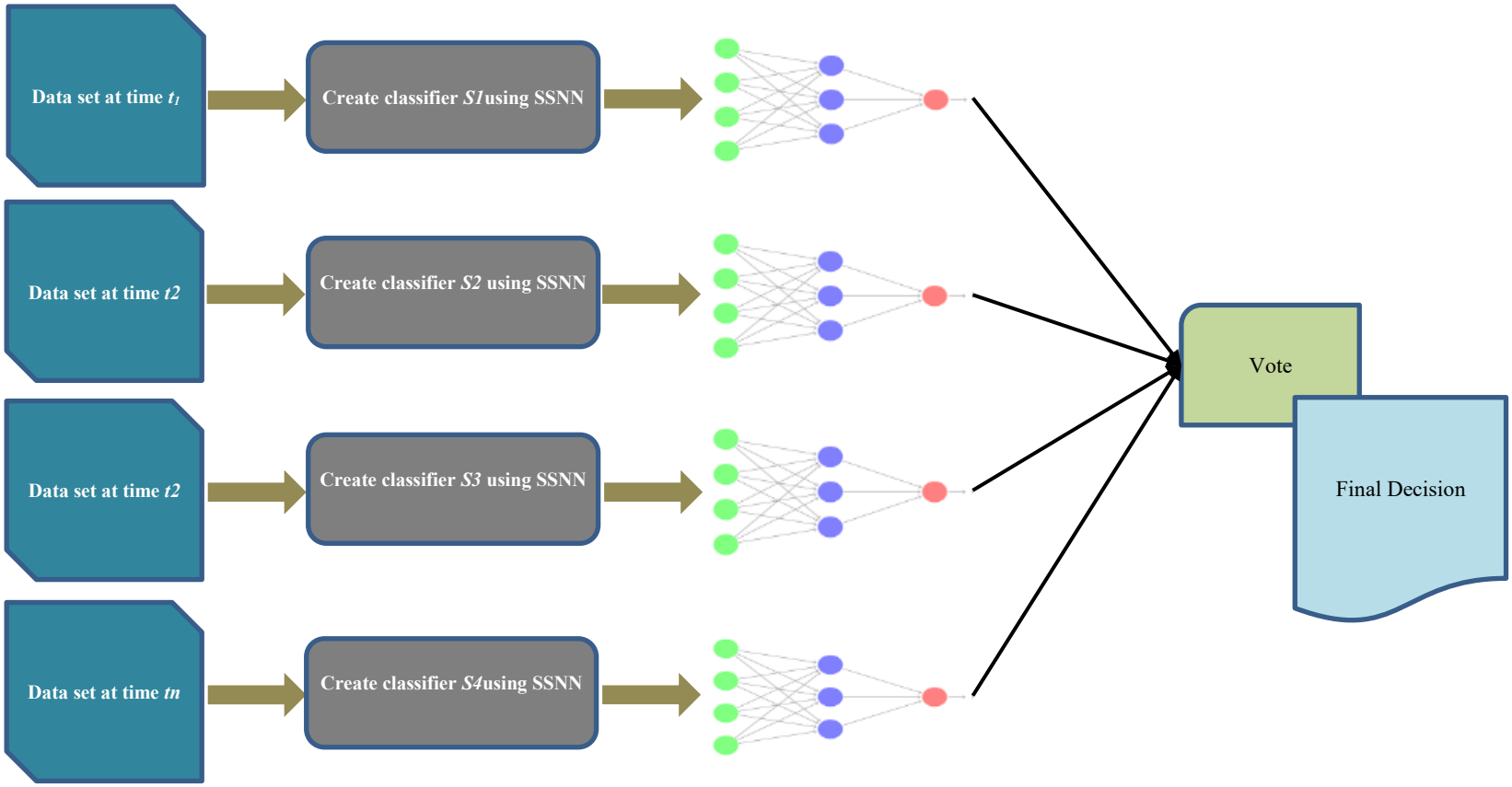


Figure 4.1 The proposed ESSNN framework

Combining multiple classifiers, i.e. ensemble of classifiers, ensures that all previously created classifiers participate in the prediction phase rather than depending only on a single classifier (in particular the one created last) in making the final class prediction. Hence, no classifier is discarded based on its age since we may face the problem of a cyclical concept drift. The proposed framework supports reinforcement learning because it uses the set of misclassified instances at time t_i to derive a new classifier at time t_{i+1}. In addition, the framework explores the advantage of regularly repeating the feature selection process to provoke new features into play.

According to section 2.4, two reasons may lead to the occurrence of the concept drift, i.e. virtual and real. The proposed framework in this chapter is more inclined to the virtual one where the features utilized in predicting the class variable may change from one data set to another. Virtual concept drift might occur unintentionally due to the changes occurring naturally in the domain. Nevertheless, it might also occur purposely with the aim to circumvent the classification model. Security based domains are good examples where virtual concept drift might be guided by some malevolent agent (Dhillon, 2001). One way for those malicious people to be the reason for the occurrence of the virtual concept drift is by hacking into the classification model, and picking a new set of features that can circumvent it as discussed in section 3.7.

Neural Network proved its superiority in several of classification domains such as image classification (Kesari et al., 2014), pattern recognition (Insung & Wang, 2007), speech recognition (Hinton et al., 2012), and medical diagnosis (Amato et al., 2013). The power of NNs is gained for several reasons, for instance, their strong ability to learn (White, 1990), their power to generalize (Hornik et al., 1989) (Basheer & Hajmeer, 2000), and their fault tolerance (McCaffrey, 2012). Yet, it might be difficult to interpret how a NN classifier

produces its results, and it is regarded as a black box (Benitez et al., 1997). Hence, even if those malevolent agents were able to hack into the NN based classification model they might face some difficulties to comprehend how the features are employed to produce the final decision. In other words, the only visible parts in any NN model are the input and output, whereas the process that transforms the inputs into outputs is obscured. This characteristic has successfully applied in different security domains such as the work done in (Arvandi et al., 2008), (Alallayah et al., 2010), and (Al-Ubaidy, 2004). Nevertheless, NN classifiers are normally created using a painstaking trial and error method. In this chapter, we have proposed an improved NN structuring algorithm called Self-Structuring Neural Network (SSNN) that simplifies structuring NN classifiers and add them to the ESSNN.

Hereinafter, we discuss the phases of the proposed framework as well as the SSNN algorithm in detail.

4.2. The main phases of the ESSNN

1. Training Data set Formation

Any supervised classification method needs a set of independent variables denoted as input features to build a classifier. Such variables have influence on dependent variable(s), i.e. class variable(s). Thus, a set of objects containing such dependent and independent variables should be collected, and from each object, a set of input features and corresponding class variable(s) are extracted. Hence, a structured training data set is created.

The resulting data set is merged with a set of instances that were misclassified by the ESSNN in previous testing phases. In other words, the newly collected data set is merged with a set of *old but may be useful* instances. The main aim of merging the set of misclassified examples with the newly collected data set is to

reinforce learning and create a strong classifier next time a new classifier is derived. One possible reason for misclassifying some instances is that the ESSNN may be lacking some knowledge. Such knowledge can be acquired by studying the misclassified examples.

The output of this step is a data set that contains all possible input features even the redundant ones; this is why we call it large data set. However, we realize that not all features are equally important in predicting the class variable (Guyon & Elisseeff, 2003). Some features might be more significant than others. Yet, the set of significant features at time t_i might become insignificant at time t_{i+1} and vice versa because of the emergence of a virtual concept drift. Hence, as soon as a large data set is formed the set of significant features are identified as we will see in section 2.

2. Creating Minimal Data sets and Concept Drift Identification

This phase aims at selecting an effective set of features from the large data set. In other words, it aims to pick a sub-set of features that most contributes in determining the value of the class variable. The output of this phase is called minimal data set since it will include only the most effective set of features whereas the redundant features will be removed. Several feature selection methods can be used to produce the minimal data set including Information Gain (IG) (Shannon, 1948), Chi-Square (Greenwood & Nikulin, 1996), Gain Ratio (Quinlan, 1993), etc. Normally, these methods assign a rank value that corresponds to the feature's significance. The set of features that have a ranking value greater than a predefined threshold value will be used to create a new classifier. However, since no agreement was found on a specific ranking value for picking the most effective set of features, several threshold values will be used to create several minimal data sets. From each resulting minimal data set,

a new classifier will be derived and the classifier that best improves the ensemble's performance is added to the ESSNN (section 3).

However, besides ensuring the simplicity and compactness of our framework, another primary aim of creating the minimal data set is to assess whether a virtual concept drift occurs or not and ensure classifiers' diversity in the resulting ensemble. To that end, we compare the features in each minimal data set with the features used in deriving the classifiers currently existing in the ESSNN.

Given a minimal dataset b, which has n distinct features A_1, A_2, …, A_n. If the same set of features in b have been used to create a classifier currently existing in the ESSNN, hence, no classifier will be created. Otherwise, we assume that a virtual concept drift might occur. In other words, there is no need to derive a new classifier if the set of features in the minimal data set has been used to derive a classifier that currently exists in the ESSNN because that would result in two classifiers having the same set of features.

However, our assumption about concept drift existence will lead us to assess to what extent the concept drift affect the overall accuracy of the ESSNN. Sometimes a virtual concept drift occurs because there are some features that have a minor impact on the ESSNN accuracy have been dropped from; or added to the minimal data set. Therefore, before the new classifier c_t is created, the accuracy of the current composite classifiers C_t is evaluated on the new large data set S_t. Hence, the new large data set is considered as a testing data set for the currently available classifiers in the ESSNN. Each classifier in the ESSNN will fetch the set of features that were used to create it, thus disregarding others and ensuring that C_t is not biased. If the overall accuracy of the combined results from C_t shows a significant drop, then a virtual concept drift occurs and

a new classifier could be added to the ESSNN. Thus, we proceed in creating a new classifier (section 3).

3. Creating a new Classifier using Self-Structuring Neural Network Algorithm

If a virtual concept drift occurs, then we use the set of minimal data sets created in section 2 to derive several classifiers. As soon as a new classifier is created, it will temporarily be added to the ensemble to assess the improvement the classifier adds to the ensemble. Then, we save the result and remove the classifier from the ensemble. The classifier that best improves the ensemble accuracy will be added permanently to the ensemble. The procedure performed to pick the best classifier is explained in Figure 4.2.

1. for each minimal data set
2. create a new classifier;
3. add the new classifier to the ensemble;
4. assess the performance of the ensemble after adding the new classifier;
5. save the result achieved in line 4;
6. remove the classifier from the ensemble;
7. end for
8. from the results achieved in line 5, pick the classifier that best improves the ensemble performance and add it permanently to the ensemble;
9. remove other classifiers;
10. remove the training data set;

Figure 4.2 An Algorithm for adding a new classifier to ensemble

The classifiers are created using the SSNN algorithm. The pseudocode of the SSNN algorithm is shown in Figure 4.3. Hereinafter, the main steps of the SSNN algorithm are discussed in detail.

```
Input
     Upload the minimal dataset and divide it to Training, Testing and Validation
     Integer Tk specifying the number of epochs
     Maximum number of possible hidden neurons
Output
     Number of neurons in the hidden layer
     Learning rate value
     Connection weights between input, hidden and output layers
Initialize
1    Number of neurons in the input layer = number of input features;
2    Number of neurons in the output layer = 1; /* Because SSNN is a binary
                                                    classification algorithm*/
3    Number of neurons in the hidden layer = 1;
4    Weights = random numbers ranging from -0.5 to +0.5;
5    Learning rate LR= 0.3;
6    momentum value = 0.2
7    Desired Error DER = 50%;
Start
8    Train the network to find the calculated error-rate CER; /*CER is calculated every
                                                        time the network is trained */
9     If CER < DER then {
10  A: Set DER= CER;
11   Train the network;
12   Update the LR after each training epoch;
13   Check the early stopping condition;
14    If (DER Achieved && iteration< Tk) then
15       Go to A;
16    Else {/* reaching the early stopping or maximum allowed number of epochs */
17       Add a new neuron to the hidden layer;
18       Train the network;
19         If (DER Achieved && iteration< Tk)
20           Go to A;
21         Else
22           Produce the Network;
23       }
24     Else (No Network can be produced)
End
```

Figure 4.3 The SSNN Algorithm Pseudocode

A. Parameter Settings

This phase involves several activities as follows:

1. The number of neurons in the input layer is set to the number of input variables offered in the training data set *(line 1 in Figure 4.3)*.
2. The number of neurons in the output layer is set to one because SSNN aims to create binary classification models *(line 2)*. In other words, we will create a single neuron that might hold two possible values.

3. The number of neurons in the hidden layer is to be determined by the algorithm. Constructive NN structuring approach normally starts with one hidden neuron (Islam et al., 2009). The SSNN algorithm creates the simplest NN structure that contains one neuron in the hidden layer *(line 3)*, this neuron is connected to all neurons in the input and output layers.
4. Small non-zero random values will be assigned for each connection weight *(line 4)*.
5. The learning rate value commonly ranges from ≈0 to ≈1 (Basheer & Hajmeer, 2000). Following the default value of WEKA (Hall et al., 2011), the learning rate and momentum value are set to 0.3 and 0.2 respectively *(line 5) and (line 6)*. However, the learning rate value will be adjusted several times during the network training process as we will see in the Main Training Phase (sub-section C).
6. One key when creating an ensemble is to derive a set of classifiers each of which produces an error-rate less than 50% (Rokach, 2010) (Dietterich, 2000) (Hansen & Salamon, 1990). Therefore, the initial desired error-rate *(DER)* is set to 50% *(line 7)*. This value will be used to assess if the algorithm can find a possible solution (sub-section B). In addition, this value will be adjusted several times, as we will see in the Main Training Phase (sub-section C).
7. The model designer specifies the maximum number of epochs.
8. The model designer sets the maximum number of allowed hidden neurons.
9. The training data set is divided into training, testing, and validation data sets. The training data set will be used to learn the model and update weights; the testing data set will be used to assess the overall performance of the derived classifier. However, the validation data set plays an important role in producing the final model, as we will see later in the Main Training Phase (sub-section C).

B. Warming-up Training Phase

In this phase *(lines 8–9)*, the algorithm decides whether to proceed with creating a new classifier or not. The algorithm computes the calculated error-rate (*CER*), aiming to determine what the *DER* to be achieved in the next training epoch is. In other words, the *CER* is the *DER* to be achieved in the next training session. Hence, the algorithm trains the network and finds the *CER*.

If the algorithm finds a *CER* less than the initial *DER* before reaching the maximum number of epochs, then the algorithm assumes that the current structure can further be improved; hence, the algorithm resets the epoch, and assumes that the *DER* to be achieved in the next training epoch is the *CER*. Then the algorithm moves to the Main Training Phase (sub-section C). On the other hand, if the *CER* is bigger than the initial *DER,* then the algorithm will be terminated *(line 24)*. The *CER* is equivalent to Mean Square Error (*MSE)* and is calculated as per equation 4.1. Where A_k is the predicted value for example *k;* and D_k is the real value associated with example *k* in a training data set having *n* examples.

$$CER = \frac{1}{n}\sum_{k=0}^{n}(A_k - D_k)^2 \qquad (4.1)$$

C. Main Training Phase

In this phase *(lines 10-13)*, the SSNN algorithm continues training the network until the *CER* is less than the *DER* or the maximum number of epochs is reached or achieving the early stopping condition.

Each training epoch starts by updating the learning rate based on the *CER* achieved in the previous epoch. One of the simplest methods for updating the learning rate is the bold driver method (Battiti, 1989). After each training epoch, the algorithm compares the *CER* at time *t* with the *CER* at time *t-1* and if the error has decreased, the learning rate is slightly increased by a specific ratio φ

in order to accelerate the error-rate reduction process and converge quickly to the possible solution. In this case, the weights are updated. On the other hand, if the error has increased or has not changed, SSNN will heavily decrease the learning rate by φ' because we might be approaching one possible solution and we need to slow down and study the possible solution more deeply. In this case, the weights are not updated. Commonly, φ and φ' are set to 0.1 and 0.5 respectively (Battiti, 1989) (Duffner & Garcia, 2007). The reason that φ is smaller than φ' is because we do not want to make a big step that causes the algorithm to diverge from the possible solution. However, as soon as the SSNN approaches a possible solution, we need to examine that solution more deeply; therefore the learning rate is heavily decreased.

After each training epoch, the SSNN algorithm calculates the error on the validation data set. When the error on the validation data set starts to increase, that mean the model has begun to over-fit the data, and the training should halt (McCaffrey, 2012). Nevertheless, the validation data set may have several local minima (as show in section 2.11); thus, if we stop training at the first increase, we may lose some points that achieve better results because the error-rate may decrease again at some further points. Therefore, SSNN algorithm tracks the error on the validation data set. If the lastly achieved error is less than the minimum achieved error, that means the generalisation ability of the classifier is improved; thus the algorithm saves the weights and continues training the network. On the other hand, if the lastly achieved error is bigger than the minimum achieved error, the algorithm continues the training process without saving the weights. However, if the lastly achieved error is bigger than the minimum achieved error with a specific threshold α, then the algorithm terminates the training process (early stopping) since exceeding that threshold value might mean that the network diverges from the ideal solution and is

difficult to converge back again. A recent study on early stopping (Riley et al., 2010) finds that the early stopping is triggered if α=50%.

For the purpose of this research, equation 4.2 has been proposed. Such equation clarifies how SSNN algorithm handles the early stopping. Where, ε is the lastly achieved error, and ε' is the minimum error.

$$IF \begin{cases} \varepsilon < \varepsilon' \xrightarrow{weights\ Saved} Continue\ training \\ (\varepsilon > \varepsilon') and (\varepsilon < [(1+\alpha) * \varepsilon']) \xrightarrow{weights\ not\ Saved} Continue\ training \\ Otherwize \rightarrow Training\ terminated \end{cases} \quad (4.2)$$

D. Improvement Phase

In the main training phase, if the SSNN obtains a *CER* less than the *DER* before reaching the maximum epoch, this could be an indication that the network can further be improved without adding any neurons to the hidden layer *(lines 14-15)*. Therefore, SSNN maintains the learning rate and weights achieved so far; resets the epoch, assumes that the *DER* to be achieved in the next training phase is the *CER*, and goes back to the Main Training Phase. Otherwise, the SSNN goes to Adding a New Hidden Neuron Phase *(line 16)*.

E. Adding a New Hidden Neuron Phase

SSNN algorithm arrives at this phase if it cannot achieve the *DER* and reaches the maximum allowed epoch or the early stopping condition. In this case, we assume that the current structure has been squeezed to the limit and the network's ability of processing the information is insufficient. Therefore, the algorithm will add a new neuron to the hidden layer *(line 17)*. The algorithm connects the new neuron to all input and output neurons, assigns small non-zero value to its weight, maintains the learning rate and weights achieved so far, and resets the epoch.

Yet, adding a new neuron to the hidden layer does not mean that this neuron is permanently added into the network, we must first assess whether the new neuron improves the network performance or not. Hence, the algorithm continues training the network *(line 18)*. If (after adding a new hidden neuron) the *DER* is achieved, then the algorithm approves adding the new neuron, maintains the learning rate and weights, resets the epoch, assumes that the *DER* to be achieved in the next training phase is the *CER*, and goes back to the Main Training Phase *(lines 19-20)*. Otherwise, the algorithm moves to Producing the Final Network Phase *(line 21)*.

The main concern in this phase is that the number of hidden neurons can freely evolve resulting in a complicated structure. Thus, the algorithm allows the system designer to set the maximum number of hidden neurons.

F. Producing the Final Network Phase

If adding a new hidden neuron to the network does not improve the network performance, then the SSNN removes the lastly added neuron, resets the learning rate and the weights as they were before adding the new neuron, terminates the training process, and the final network is produced *(line 22)*.

The most obvious network parameters to evolve in the SSNN algorithm are the learning rate, the connection weights, and the number of hidden neurons. TANH activation function has been used for the input layer, whereas, the bipolar hard limit activation function has been used for the hidden layer. The selection for the TANH activation function is that it convergence faster than other activation functions (Kriesel, 2007) (Kalman & Kwasny, 1992). However, bipolar activation function has been selected because SSNN derives binary classifiers.

4. Combine the Classifiers Results

As soon as a new unseen example arrives, each classifier in the ESSNN will come up with its own decision about the example's class and then their outcomes are combined in some ways as shown in section 2.10.2. In our ensemble, each classifier is assigned a weight based on its performance on a testing data set consisting of the testing data set achieved from the most recent data set and historical testing data sets, i.e. testing data sets of the previously created classifiers that currently exist in the ESSNN. The overall accuracy produced from classifier c_i is considered the weight of that classifier, where $i=1, 2, 3 \ldots Z$, *and* Z is the total number of classifiers in the ESSNN. When a new unseen example arrives, the ESSNN calculates the weighted voting sum for each class. The prediction value produced from the ESSNN is the class that has the highest weighted voting sum.

Finally, after creating the ensemble, any misclassified example in the testing data set will be kept and used in the next training session. Misclassified examples are used to reinforce learning and create a strong classifier next time a new classifier is created and added to the ensemble.

4.3. Chapter Summary

In this chapter, we propose a classification framework called Ensemble Self-Structuring Neural Network (ESSNN). The ESSNN has the ability to accommodate new knowledge, and does not discard previously learnt knowledge. Thus, the proposed framework furnishes a balance between stability and plasticity. The ESSNN is evaluated as soon as a new data set chunk is collected. If the produced accuracy is significantly drops, a new classifier is created and added to the ESSNN. Each classifier is assigned a

weight based on its accuracy. The prediction value produced from the ESSNN is the class that has the highest weighted voting sum.

The proposed framework is more inclined to the virtual concept drift where the significant features utilized in predicting the class variable may change over time. In this research, we explore the advantage of regularly repeating the feature selection process to provoke new features into play. The classifiers that are added to the ESSNN are created using an improved Self-Structuring Neural Networks algorithm (SSNN) that simplifies creating NN classifiers.

The SSNN algorithm will be evaluated in the next chapter (chapter 5) in order to assess if the SSNN algorithm is able to derive good classifiers and add them to the ESSNN. Later, in chapter 6, we will apply the ESSNN on a vital web security problem where a virtual concept drift might occur, that is the phishing websites classification problem with the aim to assess if it can handle the virtual concept drift and offers a balance between stability and plasticity when applied to phishing websites classification problem.

CHAPTER 5

Evaluating the SSNN Algorithm

5.1. Introduction

Creating a NN classifier is traditionally accomplished using trial and error. However, this method has several difficulties as discussed in section 2.12. In chapter 4, we have proposed an algorithm that aims at simplifying structuring NN classifiers that are added to the ESSNN framework. Such an algorithm is called Self-Structuring Neural Network (SSNN). SSNN is a family member of the constructive NN structuring approach. The classifiers derived from SSNN are assumed to learn from the training data set and to generalise the output on test data set.

In this chapter, several experiments will be accomplished to evaluate the performance of the SSNN algorithm with the aim to assess if the SSNN is able to derive good classifiers. Two sets of experiments will be conducted. The first set compares the SSNN against several DM classification algorithms on a number of binary data sets from the UCI repository. In the second set of experiments, the SSNN will be applied to phishing websites classification problem and the results will be compared to the results achieved from several other DM classification algorithms.

5.2. Evaluating the SSNN on UCI Data sets

5.2.1. Algorithms Used for Comparison and the Experimental Settings

The experimental evaluation compares the SSNN algorithm with Decision Tree (C4.5) (Quinlan, 1998), Bayesian Network (BN) (Friedman et al., 1997), Logistic Regression (LR) (Witten et al., 2011), and the standard Feed Forward Neural Network (FFNN) algorithm implemented in WEKA (Hall et al., 2011). The selection of these algorithms is because these algorithms utilise different strategies in producing classifiers and are commonly used in creating classification models (Abdelhamid et al., 2012), (Thabtah et al., 2005), (Witten et al., 2011). Such algorithms have been discussed briefly in sections Appendix A. However, the FFNN algorithm assumes that the number of neurons in the input layers equals the number of attributes in the training data set, whereas the number of neurons in the output layer equals the number of classes. The number of neurons in the hidden layer is the average number of neurons in the input and output layers (Hall et al., 2011), and is calculated as per equation 5.1. However, the default epoch size used in FFNN is 500 (Hall et al., 2011).

$$number\ of\ hidden\ neurons = \frac{number\ of\ input\ neurons + number\ of\ output\ neurons}{number\ of\ layers} \qquad (5.1)$$

Other algorithms were selected since they use different strategies in producing classification models and they have been discussed briefly in sections Appendix A.

For all comparable algorithms, we use the default parameter settings of WEKA (Hall et al., 2011). Whereas, for SSNN algorithm, two input values should be entered from the system designer, i.e. number of epochs and maximum number of possible hidden neurons. There is no rule of thumb in which one can decide on these values (Basheer & Hajmeer, 2000). Therefore, following

some recent studies which employ NN to create classification models in different domains (Kesari et al., 2014), (Madhusmita et al., 2012), (Ganatra et al., 2011), (Insung & Wang, 2007), (Hossein et al., 2003), (Shamik et al., 2011), (Paulin & Santhakumaran, 2011), (Amato et al., 2013) we set the maximum number of possible neurons in the hidden neurons to 15. Yet, these studies utilise different epoch sizes and the most commonly used epoch size values are 100, 200, 500, and 1000. Four sub-experiments will be conducted, in which the maximum number of possible hidden neurons is 15, and epoch size has been set to 100, 200, 500, and 1000 for sub-experiments 1, 2, 3, and 4 respectively. The SSNN algorithm has been implemented in Java. All experiments were conducted in a system with CPU Pentium Intel® CoreTM i5-2430M @ 2.40 GHz, RAM 4.00 GB. The platform is Windows 7 64-bit Operating System.

5.2.2. Training Data sets

Ten different binary classification data sets from the University of California Irvine repository (UCI) (Lichman, 2013) have been chosen. We have picked small, medium and large data sets with several numbers of attributes for fair selection. Table 5.1 shows their description, i.e. name, number of attributes, number of instances, and class distribution for each data set.

Table 5.1 UCI data sets description

Data set	Number of attributes	Number of Instances	Class Distribution
Breast	9	699	66% 34%
Labor	16	57	35% 65%
Liver	6	345	42% 58%
Ionosphere	34	351	36% 64%
Pima	8	786	65% 35%
Tic-Tac	9	958	35% 65%
Sonar	60	208	47% 53%
Hepatitis	19	155	21% 79%
Vote	16	435	61% 39%
kr-vs-kp	36	3196	52% 48%

Some data sets hold categorical values. Yet, SSNN algorithm requires converting these categorical values into numerical values. In addition, SSNN

algorithm will produce either 1 or -1 because the activation function in the hidden layer is a bipolar hard limit activation function. Hence, any class variable holds values other than 1 and -1 will be converted to 1 and -1 for experiments purposes. In order, to differentiate input attributes from class variables, we assign 0.5 and -0.5 for the binary attributes, and 0.5, 0, and -0.5 for trinary attributes. Other attributes holding numerical values are left without any changes.

5.2.3. Validation Technique and Evaluation Metrics

The hold-out validation technique is used in our experiments. Following some previous studies (Bubtiena et al., 2011), (McCaffrey, 2012), (Sivanandam et al., 2006) (Amato et al., 2013), (Gabralla et al., 2014), all data sets will be divided into 80% for training and 20% for testing. Moreover, when creating the SSNN classifiers the training data sets will be further divided into 80% for training and 20% for validation.

In any supervised classification model, four classification possibilities are employed as per confusion matrix shown in Table 5.2.

Table 5.2 Confusion Matrix

Predicted Value	Actual Value	
	Classified as Positive	Classified as Negative
Is Positive	*TP*	*FP*
Is Negative	*FN*	*TN*

Where True Positive (Sensitivity) (TP) is the number of positive examples correctly classified as positive, False Negative (FN) is the number of positive examples incorrectly classified as negative, False Positive (FP) is the number of negative examples incorrectly classified as positive and True Negative (Specificity) (TN) is the number of negative examples correctly classified as

negative. Several evaluation metrics can be derived from the confusion matrix. In the first set of experiments, the SSNN algorithm will be compared with other classification algorithms in terms of produced error-rate (described in section 5.2.5), relative accuracy-rate (section 5.2.6), and time taken to produce classifiers (section 5.2.7) these metrics are commonly used to evaluate any classification model (Abdelhamid et al., 2012) (Thabtah et al., 2005).

5.2.4. Results and Discussion

1- Error Rate Analysis

The error rate is calculated as per equation 5.2.

$$error\text{-}rate = 1 - Accuracy(\%) \quad (5.2)$$

Where *Accuracy(%)* is the proportion of instances that have correct classification from the size of the data set and is calculated as per equation 5.3.

$$Accuracy\ (\%) = \frac{TP+TN}{TP+FP+TN+FN} \quad (5.3)$$

Table 5.3 shows the error rate produced from SSNN and other considered algorithms.

Table 5.3 Error rate of SSNN and other considered algorithms

Data set	C4.5	BN	LR	FFNN	SSNN-100	SSNN-200	SSNN-500	SSNN-1000
Breast	7.14%	3.57%	**2.86%**	4.29%	3.57%	3.57%	3.57%	3.57%
Labor	27.27%	9.09%	9.09%	9.09%	9.09%	9.09%	**9.09%**	9.09%
Liver	40.58%	39.13%	**28.99%**	31.88%	39.13%	31.88%	34.78%	34.78%
Ionosphere	20.00%	**12.86%**	18.57%	15.71%	20.00%	17.14%	17.14%	15.71%
Pima	24.03%	**17.53%**	18.83%	25.97%	19.48%	18.18%	20.13%	20.13%
Tic-Tac	16.15%	29.69%	1.57%	3.15%	2.08%	3.15%	**1.44%**	3.56%
Sonar	19.05%	23.81%	16.67%	9.52%	14.29%	11.52%	**9.52%**	9.52%
Hepatitis	32.26%	9.68%	16.12%	16.12%	16.12%	19.35%	**9.68%**	16.12%
Vote	5.75%	11.49%	**1.15%**	4.60%	3.45%	4.60%	4.60%	3.45%
kr-vs-kp	0.47%	12.52%	2.19%	0.78%	2.03%	0.47%	**0.47%**	0.47%
Average	**19.27%**	**16.94%**	**11.61%**	**12.11%**	**12.92%**	**11.90%**	**11.04%**	**11.64%**

From Table 5.3 and Figure 5.1, the results clearly show that the classifiers produced from the SSNN when epoch size is set to 500 have the least error-rate

on average. Such classifiers have achieved on average 0.57%, 1.07%, 5.90% and 8.23% lower error-rate than LR, FFNN, BN and C4.5 respectively. These results have a good sign that NN based algorithms, i.e. FFNN and SSNN generate better classifiers than C4.5 and BN in terms of error-rate. One possible reason is the fact that these algorithms have the ability to deal with fault tolerance situations, i.e. their performance is not significantly affected if some data are missing in the training data set (Kantardzic, 2011). That explains the high error-rate of C4.5 in some data sets such as Breast, Labor, and Hepatitis given that these data sets have 16, 326, and 167 missing values respectively. The average error-rate produced from LR is slightly better than the classifiers produced from SSNN when epoch size is set to 100, 200 and 1000, given that LR produces good results if it is applied in binary classification domains (Witten et al., 2011). That explains the low error-rate produced from LR on Vote data set since in this data set not only the class variable holds binary values but also the input features. SSNN outperforms all considered algorithms in terms of average error-rate when epoch size is set to 500.

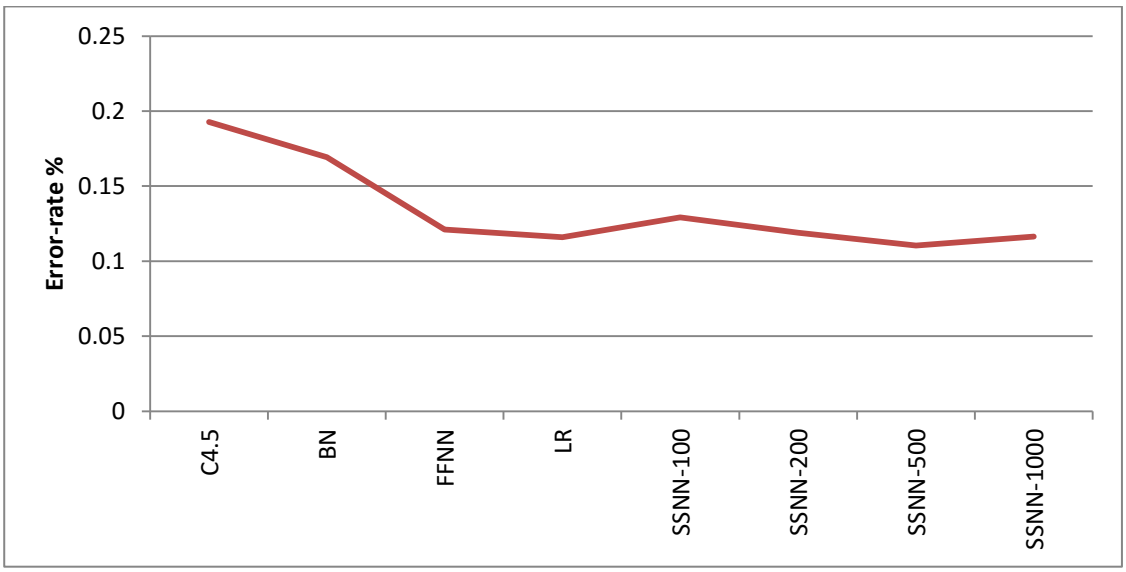


Figure 5.1 Error-rate from SSNN and other considered algorithms

Overall, SSNN is able to produce competitive results on different epoch sizes. That can be attributed because SSNN is able to produce well-structured classifiers. To elaborate, SSNN keeps pushing on the learning rate to improve the NN performance by adjusting the desired error-rate several times before adding a new hidden neuron. The learning rate in SSNN is not static but it changes after each training epoch. If the error-rate achieved at epoch *i* is smaller than the error at epoch *i-1*, the SSNN increases the learning rate in order to converge quickly to the possible solutions. However, if the error-rate at epoch *i* is higher than the error-rate at epoch *i-1* the SSNN decreases the learning rate since we might approach one possible solution and the SSNN needs to slow down and deeply study that solution. Yet, if the SSNN cannot achieve the desired error-rate and it reaches the maximum number of epochs then the SSNN adds a new hidden neuron. In addition, the mechanism of producing the weights in SSNN might be another reason for deriving good classifiers since the weights are only saved when the performance of the network on the validation data set has improved. However, the FFNN algorithm produces the weights that have been achieved in the last training epoch. To illustrate further on the error-rates achieved by all algorithms, Table 5.3 above demonstrates the data sets we considered and the error-rate values of each algorithm. It should be mentioned that in order to derive general conclusions on the performance of SSNN, we choose large (kr-vs-kp), medium (Tic-Tac), and small (Labor) data sets.

2- Statistical Significance Comparison

In order to determine the significance in performance, we performed a paired *t*-test between the results of the SSNN and other considered algorithms using the WEKA platform (Hall et al., 2011). Paired *t*-test is a statistical method for comparing the result of measuring one set of data twice, i.e. with two different algorithms. The comparison is based on the overall accuracy over the 10

selected datasets. Table 5.4 shows the comparison results between the SSNN and other considered algorithms with 95% confidence level in which *W* signifies how many times the SSNN won, and *L* signifies how many times the SSNN losses. From the results we can clearly see that the SSNN achieved a significant improvement over C4.5 on most data sets (9 wins and 0 losses) when the epoch size is set to 200, 500, and 1000. The SSNN also produced a significant improvement over BN and FFNN (5 wins and 2 losses) when the epoch size is set to 500. However, at the same epoch size, the SSNN produced comparable results (5 wins and 4 losses) when compared to the LR algorithm.

Table 5.4 A statistical significance comparison of SSNN against other algorithms

Algorithm	C4.5		BN		LR		FFNN	
	W	***L***	***W***	***L***	***W***	***L***	***W***	***L***
SSNN-100	8	1	4	3	3	6	4	4
SSNN-200	9	0	5	3	4	5	3	3
SSNN-500	9	0	5	2	5	4	5	2
SSNN-1000	9	0	5	3	4	5	4	2

3- Relative Accuracy Rate Analysis

Figures 5.2 - 5.5 show the relative accuracy rate (RAR) that represents the difference in accuracy rate of SSNN when epoch size equals 500 (best classifiers produced from SSNN) compared with those resulted from C4.5, BN, LR, and FFNN. RAR denotes how much bad or good SSNN behaves when compared to C4.5, BN, LR, and FFNN on the considered data sets. The RAR is calculated as per equation 5.4.

$$RAR = \frac{(accuracy\ rate_{SSNN}) - (accuracy\ rate_{Target})}{(accuracy\ rate_{Target})} \qquad (5.4)$$

From the results, we find that the RAR of C4.5 is positive on 9 out of 10 data sets since SSNN achieves higher accuracy-rate than C4.5 as seen in Figure 5.2. In Figure 5.3, we can see that the RAR is negative in 2 cases only, i.e. in Pima and Ionosphere data sets, because the SSNN produces lower accuracy than BN

in these two data sets. However, in Figure 5.4, the RAR on Breast, Liver, Pima and Vote data sets is negative since SSNN achieves lower accuracy-rate than LR on these data sets. In Figure 5.5, the RAR is negative in two cases only, i.e. on Liver and Ionosphere. Overall, SSNN produces higher RAR in most cases. That can be attributed because SSNN is able to produce well-structured classifiers in terms of number of hidden neurons and weights space.

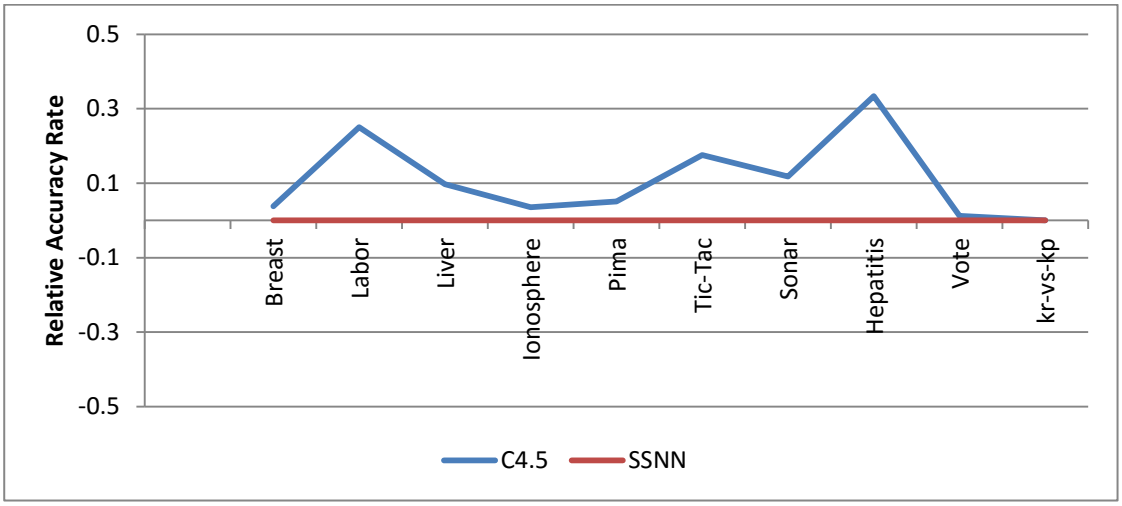


Figure 5.2 RAR (C4.5 and SSNN)

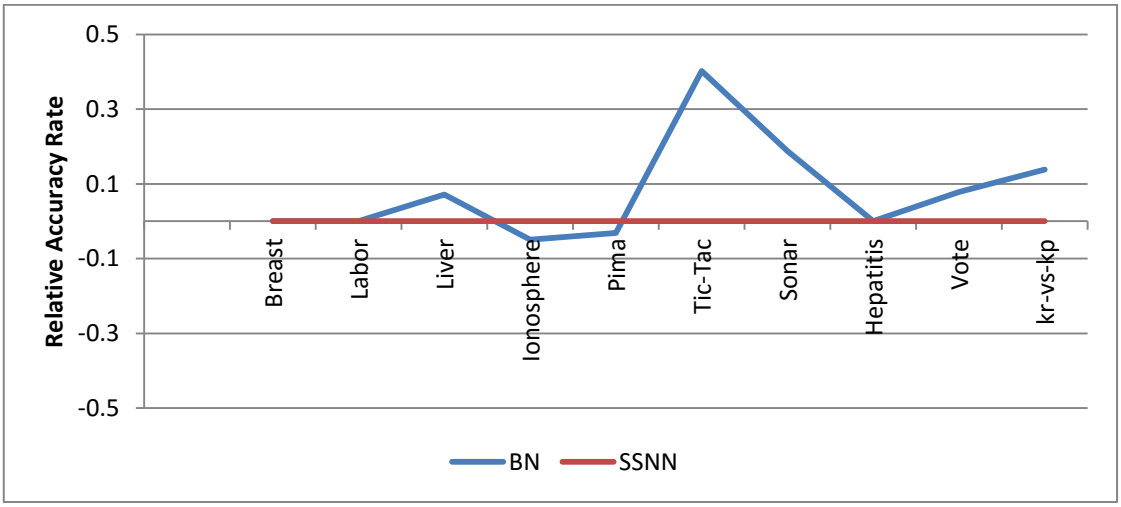


Figure 5.3 RAR (BN and SSNN)

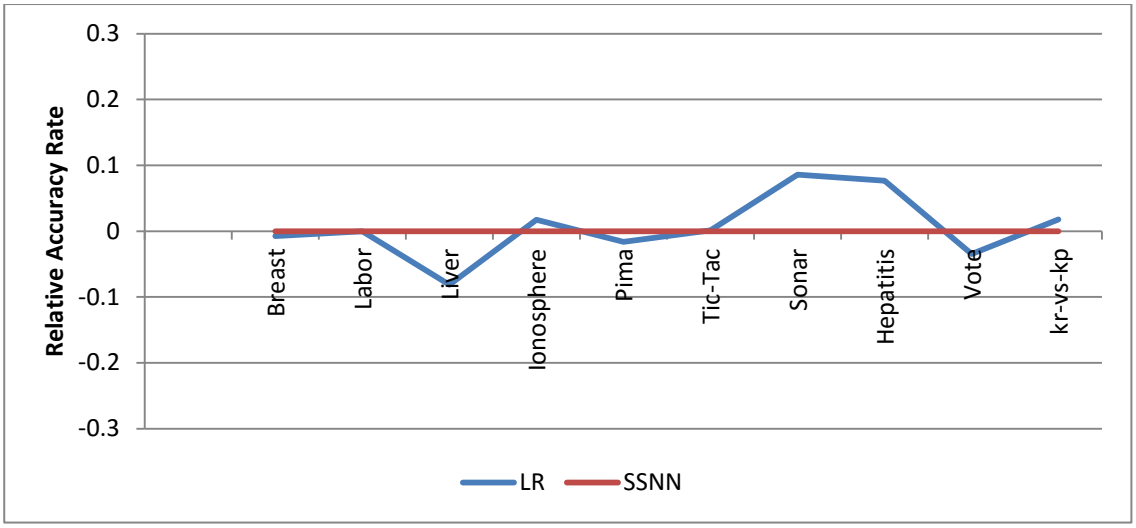


Figure 5.4 RAR (LR and SSNN)

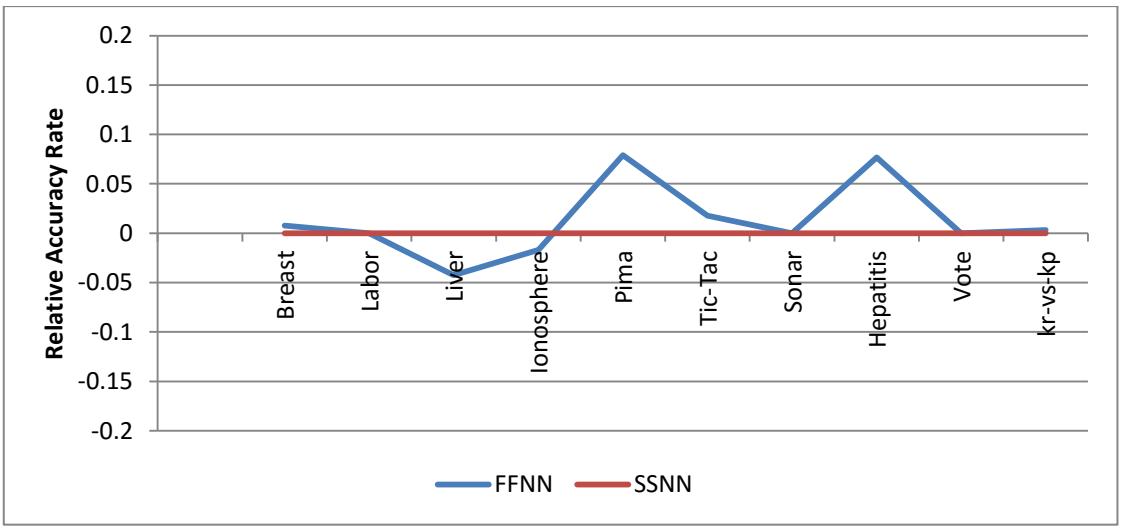


Figure 5.5 RAR (FFNN and SSNN)

4- Processing Time Analysis

The processing time needed to derive the classifiers in SSNN when epoch size is set to 500 (best classifiers produced from SSNN) has been contrasted with those of C4.5, BN, LR, and FFNN.

Table 5.5 shows the time (in seconds) needed for SSNN and other considered algorithms. The average time needed for SSNN to derive classifiers is higher than the average time needed for FFNN. That can be because SSNN performs several sub-trainings in each training session before producing the final classifier. In other words, SSNN not only trains the network but also it makes

changes on the network structure. However, the average time needed for FFNN is just for a single training session only where the network structure is static. Nevertheless, in practice, the system designer has to perform several training sessions with several parameter settings to decide on the best structure.

In general, the time needed for NN based algorithms, i.e. FFNN and SSNN is longer than other considered algorithms. This is due to the repeated process of updating the connection weights that requires doing several mathematical calculations.

Table 5.5 Time needed to build the model (in seconds)

Data set	C4.5	BN	LR	FFNN	SSNN-500
Breast	0.01	0.03	0.01	0.78	1.18
Labor	0.01	0.01	0.02	0.31	0.53
Vote	0.03	0.01	0.08	0.67	1.49
Pima	0.06	0.04	0.03	0.58	1.32
Tic-Tac	0.03	0.01	0.12	3.41	5.47
Liver	0.02	0.01	0.01	0.24	1.08
Sonar	0.03	0.01	0.06	3.04	5.56
Hepatitis	0.01	0.01	0.04	0.41	1.01
Ionosphere	0.03	0.01	0.09	2.23	5.44
kr-vs-kp	0.02	0.01	0.03	25.75	36.54
Average time	**0.025**	**0.015**	**0.049**	**3.742**	**5.962**

5.3. Evaluating the SSNN algorithm on Phishing Data Set

5.3.1. Experimental Settings

This set of experiments evaluates the applicability of the SSNN algorithm to phishing websites classification problem and the results will be compared against the same set of classification algorithms shown in section 5.2.1. For all comparable algorithms as well as for the SSNN algorithm, we will use the same parameter settings as shown in section 5.2.1.

Three different feature selection methods will be used including Information Gain (Shannon, 1948), Chi-Square (Greenwood & Nikulin, 1996) and Gain Ratio (Quinlan, 1993) with the aim to empirically assess these methods and their effect on the performance of the SSNN and other comparable classification algorithms. The selection for these methods is because they are commonly used for feature selection in the domain of phishing websites classification (Pan & Ding, 2006), (Ma et al., 2009), (Aburrous et al., 2010 B), (Basnet et al., 2012), (Abdelhamid et al., 2014), (Mansour & Alshihri, 2015).

5.3.2. Validation Technique and Evaluation Metrics

In this set of experiments, we will use the same validation technique discussed in section 5.2.3. Following previous studies related to phishing classification (Abu-Nimeh et al., 2007), (Zhang et al., 2007), (Fette et al., 2007), (Miyamoto et al., 2008), (Basnet et al., 2008), (Aburrous et al., 2010 C), (Abdelhamid et al., 2014) and (Mansour & Alshihri, 2015) we use a set of evaluation metrics that can be derived from the confusion matrix shown in section 5.2.3. These evaluation metrics are as follows:

1. Precision (P): the rate of correctly classified legitimate websites in relation to all instances that are classified as legitimate and is calculated as per equation 5.5.

$$P = \frac{TP}{TP+FP} \qquad (5.5)$$

2. Recall (R): is equivalent to TPR (Sensitivity).
3. F1-score (Harmonic Mean): is the weighted average of P and R. F1-score takes both FP and FN into account and is calculated as per equation 5.6. This metric weights R and P equally, and a good classifier will maximize both P and R simultaneously. Thus, moderately good performance on both will be favoured over good performance on one and poor performance on the other.

$$F1 = \frac{2\,P\,R}{P+R} \quad (5.6)$$

4. Accuracy (ACC): the overall rate of correctly classified legitimate and phishing websites in relation to the total number of instances in the testing data set and is calculated as per equation 5.7.

$$Acc = \frac{TP+TN}{TP+FP+TN+FN} \quad (5.7)$$

Where TP is the number of legitimate websites correctly classified as legitimate, TN is the number of phishing websites correctly classified as phishing, FP is the number of legitimate websites incorrectly classified as phishing, and FN is the number of phishing websites incorrectly classified as legitimate.

5.3.3. Significant Features across Phishing Websites

Nowadays, phishers employ a myriad of features to create phishing websites and an air of temptation to mislead honest users. James substantiated this argument in his book (James, 2005) by suggesting that phishers use a set of common features in designing phishing websites to strengthen their deceptive tactics and selfish objectives. Depending on a more inclusive set of features, a DM model would certainly produce a better performance. Yet, the set of effective features might differ from time to time (Ma et al., 2009) (Basnet et al., 2012).

In order to collect the distinctive features of phishing websites, we recognize that there are two possible ways. One of which is by reviewing previous studies; since scholars who have used such features in their studies have already confirmed that these features commonly occur in phishing websites. However, in this research we followed one more route by analysing a set of phishing and legitimate websites to determine the differences between an authentic and a fake one. To that end, a data set consisting of 2,604 legitimate websites and 2,509 phishing ones have been collected. The data set collection

methodology has been described in detail in section 5.3.4. We have divided phishing website features into four groups as per Figure 5.6.

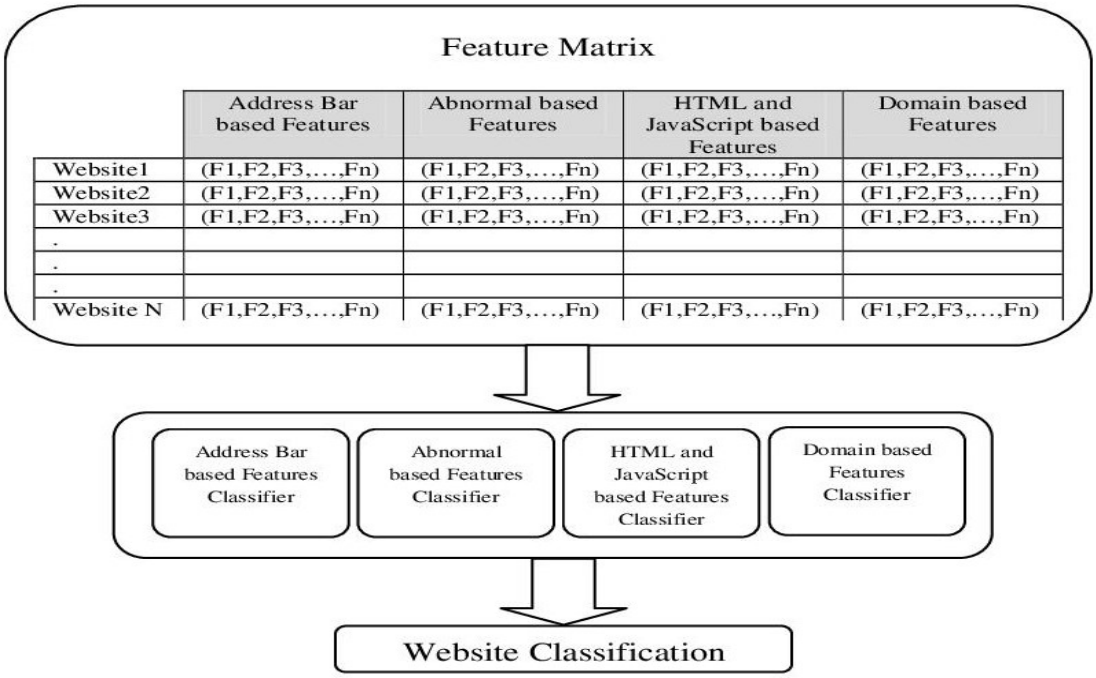


Figure 5.6 Categorizing Phishing website features

We managed to collect 30 features. Later, in the experimental section we use Information Gain to assess these features.

5.3.3.1. Address Bar based Features

1- Using the IP Address

If an IP address is used as an alternative of the domain name in the URL, such as (*http://125.98.3.123/fake.html*), users are excused if they doubt that someone is trying to steal their personal information (Aburrous et al., 2010 C). Sometimes the IP address is even converted into hexadecimal code as shown in the following link (*http:// 0xCA. 0x62. 0xCC. 0x58 /3/paypal.uk/index.htm*). To extract this feature, we examine whether the domain part of the URL (Figure 5.7) contains an IP address or not.

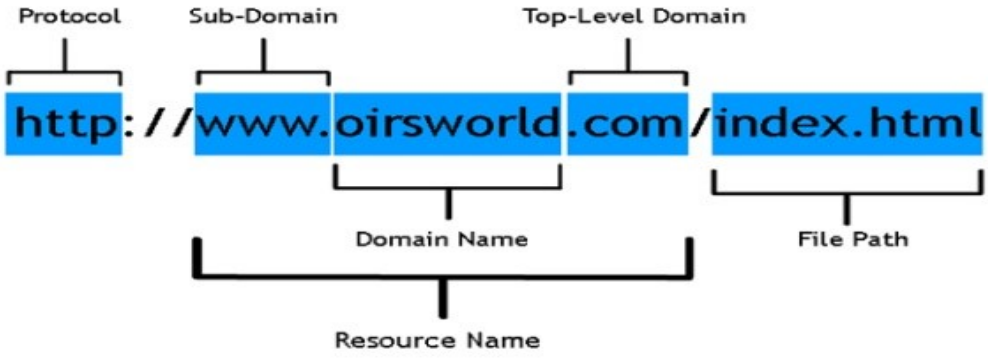


Figure 5.7 URL anatomy

2- **Long URLs**

Phishers might use long URLs to hide the suspicious part in the address bar. For example:

*http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8****@phishing.website.html***

According to (Basnet et al., 2011) the maximum length of genuine URL is 75 characters. However, the authors did not clarify the reasons behind their chosen value. In another study (McGrath & Gupta, 2008), it has been stated that URL lengths peak at 22 characters for legitimate websites in the Directory Mozilla[6] (DMOZ) (Skrenta & Truel, 2011). However, the same study revealed that there are 67 characters for the phishing URLs in PhishTank (PhishTank, 2011) and 107 for the phishing URLs in MarkMonitor (MarkMonitor, 2013). However, phishers are nowadays utilizing smaller URLs in an attempt to add an atmosphere of legality to their hostile links. In general, there is no agreement on a reliable length that separates phishing websites from legitimate ones.

[6] A multilingual open-content directory.

3- **Using URL Shortening Services (Tiny URL)**

URL shortening is a method on the World Wide Web in which a URL may be made considerably smaller in length and still leads to the required webpage. For example, a link such as *http://portal.hud.ac.uk/* can be shortened to *bit.ly/19DXSk4*. However, it is abnormal to find a legitimate website using shortening service. By looking into our data set, we found 107 TinyURLs and none of them was found in the legitimate websites data set.

4- **URL with @ Symbol**

One technique commonly used to lure users is by utilizing the @ symbol. The existence of such a symbol in the URL leads the browser to ignore everything preceding the symbol, since the real address often follows the @ symbol. After reviewing our data set, we found 90 URLs having the @ symbol and all of them were found in the phishing data set.

5- ***.htaccess*** **Redirecting**

The Hypertext Access (*.htaccess*) is a configuration file used to modify the configuration of the Apache Web Server (Starr, 2012). One of these configurations is the redirect functionality. Phishers can use *.htaccess* to redirect users to a phishing webpage. The *.htaccess* looks for any request for a specific webpage and if it finds that request, it forwards the user to a new webpage. For example, if a phisher wants to redirect users from *legitimate.html* to *phishing.html* he can use the following syntax:

redirect legitimate.html http://www.fake.ca/phishing.html

Any user visiting *legitimate.html* will end up on *http://www.fake.ca/phishing.html*. However, this technique results in adding double slash (//) at the beginning of the redirected webpage as seen in the following example:

http://www.legitimate.html//http://www.phishing.html

We will check the position where the // appears. We have found that if the URL starts with HTTP then the // normally appears in the sixth position. However, if the URL uses HTTPS then the // appears in seventh position. Otherwise, the // will cause the URL to be transferred to another webpage.

6- **Adding Prefix or Suffix Separated by (-) to the Domain**

The dash symbol is infrequently used in genuine URLs (Aburrous et al., 2010 B). Phishers resort to adding prefixes or suffixes separated by (**-**) to the domain name so that users feel that they are dealing with a legitimate webpage. For example: *http://www.Confirme-paypal.com/*.

7- **Sub-Domain and Multi Sub-Domains**

Another technique used by phishers to scam users is by adding multi sub-domains to the URL (Gastellier-Prevos et al., 2011) (Leyden, 2011) as in the following URL: *http://sigin.ehay.it.ws.ebadell.it.vividsong.com/*.

8- **HTTPS**

HTTPS is a Hyper Text Transfer Protocol (HTTP) plus Secure Sockets Layer (SSL). Usually, legitimate websites utilise secure domains every time sensitive information is transferred. SSL indicates that the clients are connected to the server they presume and not to any other third party. In addition, SSL encrypts the network traffic, therefore nobody other than the client and server can eavesdrop. The presence of HTTPS is an important sign of website validity; nevertheless this is clearly not enough. For instance, in 2005, Netcraft Toolbar Community identified more than 450 phishing URLs utilising fake HTTPS (Miller, 2005). Fake certificates can be utilised to trick users into thinking a

malicious website is legitimate. Therefore, we further need to check the certificate assigned with HTTPS, including the certificate age, and the extent of a trust certificate issuer. Certificate authorities that are frequently listed among the top trustworthy names include (Best SSL Certificates, 2011): *GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster and VeriSign*. However, by testing out our data sets, we have found that the minimum age of a reputable certificate is two years. In our phishing data set, we find 2321 URLs that do not support HTTPS, whereas in legitimate data sets we found 115 items. Unlike previous researches that assume that a URL is valid if it is using an HTTPS protocol, we further examine the certificate authority provider as well as the certificate age.

9- **Using Free Hosting Domains**

It is abnormal to find a legitimate websites hosted on a free domain-hosting server. Commonly, reputable companies pay for web hosting service providers to maintain their domain name. In a recent report (Aaron & Manning, 2014 B), it has been shown that 25% of phishing websites identified in the first half of 2014 are using free domain-hosting servers. Information about hosting providers can be obtained from WHOIS database (WHOIS, 2005).

10- **Favicon**

A favicon is a graphic image (icon) associated with a specific webpage. Many organisations show favicon as a visual reminder of the website identity in the address bar as per Figure 5.8. In our data sets, we have found 139 phishing websites using such feature. None of them loads the favicon from the domain shown in the address bar but from the genuine domain. This could be attributed to the fact that most phishing webpages are simple copies of genuine ones.

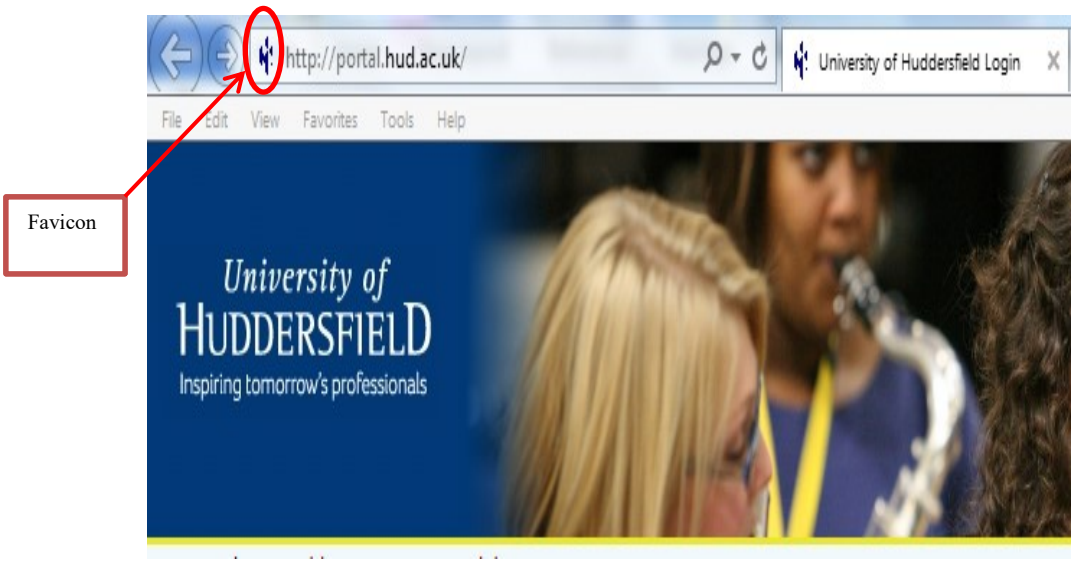


Figure 5.8 Webpage where favicon appears

11- **Using Non-Standard Port**

Port 80 and port 443 are the default ports for HTTP and HTTPS respectively[7]. This implies that no need for the user to specify the port while he browsing the Internet. However, phishers tend to add a port number to the URL in order to bypass the security detection tools that may monitor a particular port number. In our data set, we have found 28 URLs having a port number other than the default ports, all of them were found in the phishing data set.

12- **The Existence of (HTTPS) Token in the Domain Part**

Several improvements have been made on Firefox since the seventh version was released. In addition to other improvements, one of the changes made is that Firefox no longer displays the (HTTP://) part of a URL. For instance, when visiting (*http://portal.hud.ac.uk/*), Firefox will show (*portal.hud.ac.uk*) in the URL address bar as shown in Figure 5.9.

[7] A full list of the common ports can be achieved from:
http://www.networksorcery.com/enp/protocol/ip/ports00000.htm

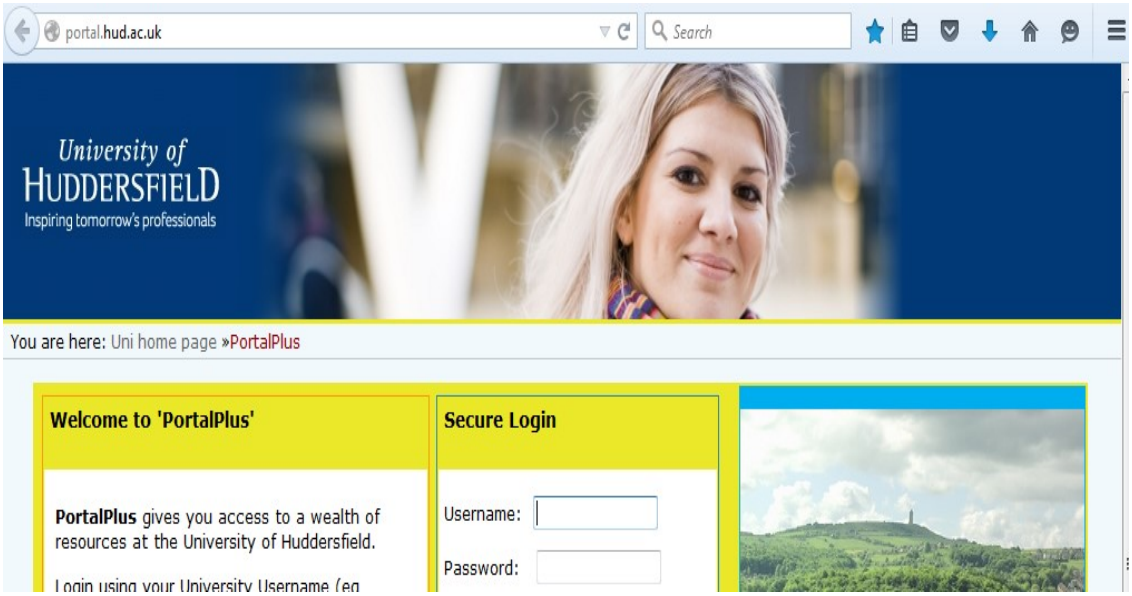


Figure 5.9 Firefox Does not Show the http protocol

However, Firefox is still showing (HTTPS://) in the URL address bar when visiting a website that uses the HTTPS protocol. The phishers were aware of such behaviour, thus they might add the HTTPS token to the URL in order to trick users. For example, if we type the following URL in the Firefox address bar (*http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/*), the shown part of this URL is (*https-www-paypal-it-webapps-mpp-home.soft-hair.com*), therefore, if HTTPS is used in the domain part of the URL that could be a sign that the URL belongs to phishing attempts.

5.3.3.2. Abnormal based Features

1- **Request URL**

When designing their phishing webpage, phishers might modify a minimal part of the targeted legitimate webpage and maintain all other links dispatched to the real webpage. This is most likely because mirroring tools are an easier way to setup a phishing website. Request URL examines whether the objects contained within a webpage such as images, videos and sounds are loaded

from another domains (Pan & Ding, 2006). In legitimate webpages, the webpage address and most of objects embedded within the webpage are sharing the same domain. For example, suppose that the URL typed in the address bar is (*http://www.hud.ac.uk/students/*), hence, we extract the link associated with the keyword <*src*=> from the webpage source code and check whether the domain in the address bar is different from that in <*src*> tags. For this feature, we calculate the rate of URLs in the webpage source code that have different domain other than the domain shown in the address bar. We compute the feature existence rate, not the frequency because the number of request URLs fluctuates from one webpage to another.

2- **URL of Anchor**

An anchor is an element defined by the < *a* > tag. For this feature, we examine:

1. If the website and the anchor tag have different domains
2. If the anchor is not linked to any webpage (Aburrous et al., 2010 C), such as: <a href="#">, <a href="#content">, <a href="#skip"> or <a href="JavaScript ::void(0)">

Same as Request URL, we calculated the feature existence rate.

3- **Links in <Meta>, <Script> and <Link> Tags**

Commonly, legitimate websites to use < *Meta* > tags to offer metadata about the HTML document; < *Script* > tags to create a client side scripts; and < Link > tags to retrieve other web resources. It is expected that these tags are linked to the same domain of the webpage. By reviewing the phishing websites data set, we have found great portions of these tags are linked to the genuine domain. For this feature, we calculate the existence rate.

4- **Suspicious Action Upon Submitted Information**

As soon as the user submitted his information, an action should be taken upon this information. Usually, this task is accomplished using the so-called Server Form Handler (SFH). A form handler is a program that runs on the web server that takes the information entered in the HTML form and does something with it. If the SFH contains an empty string or (*about:blank)*, hence, the webpage is considered doubtful (Aburrous et al., 2010 B). In our phishing data set, we have found 101 webpages containing either an empty SFH or a SFH pointing to a domain different to what is shown in the browser address bar. However, no legitimate websites have empty or *about:blank* SFHs.

5- **Submitting Information to Email**

Web forms allow users to submit their personal information to a server for further processing. A phisher might redirect the user's information to his personal email. To that end, server side programming language might be used to redirect the user's information using *mail()* function. However, a client side function might also be used for redirecting user's information using the *mailto:* function. In our data set, we found 127 data set items in which the user's information sent to a personal email, none of these items was found in the legitimate data set.

6- **Website's Owner**

In any secure website, the digital certificate provides information about the website's owner (Jalal & Peter, 1999). Reviewing our data set, we found that for phishing websites, the owner field does not provide any information as per Figure 5.10. On the other hand, the owner field in secure legitimate websites is clearly provided as per Figure 5.11.

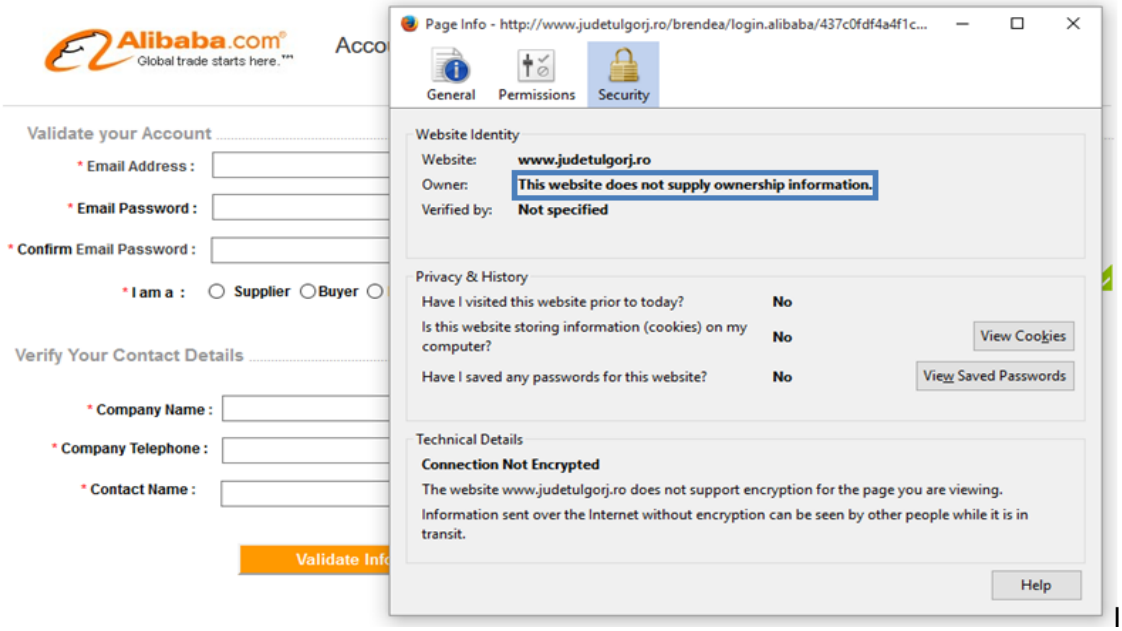


Figure 5.10 A URL not supporting information about the owner

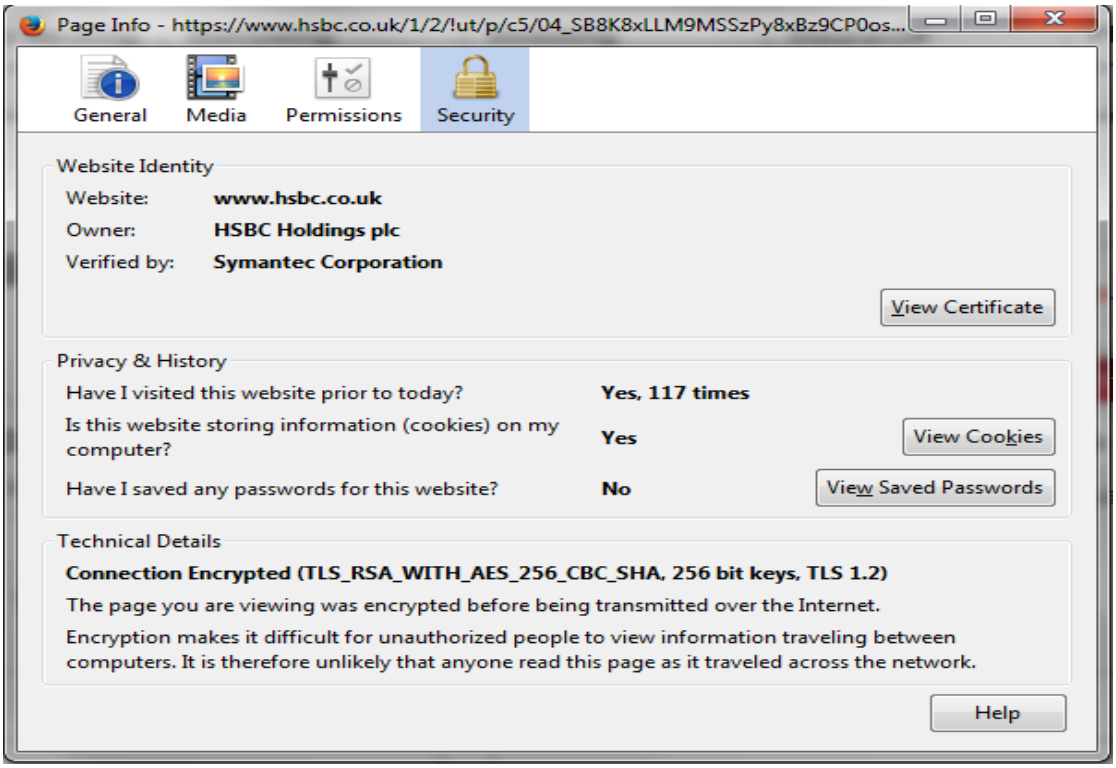


Figure 5.11 A secure website clearly supporting information about the owner

5.3.3.3. HTML and JavaScript based Features

1- **Redirect Page**

Phishers might use this feature with the intention of hiding the real link and asking the users to submit their information to a fake webpage. In a recent study (Aburrous et al., 2010 C), it has been found that phishing websites that have this feature are redirected three or more times. However, the same study revealed that some legitimate websites might also have this feature but they are redirected for one time only. The (*window.location*) object is commonly used to redirect the users to another webpage as shown in the following example: *window.location = http://www.fakewebsite.cc;*

2- **Status Bar Customization**

Phishers may use JavaScript to show the URL of the genuine website in the status bar as soon as the mouse comes across some hot areas on the fake webpage such as the register button and the signin button (Pan & Ding, 2006). To extract this feature, we examine the webpage source code, particularly the (*onMouseOver)* event, and check whether it shows a URL with a different domain than that is shown in the address bar.

3- **Disabling Right Click**

Phishers may use JavaScript to disable the right click function so that users cannot view the webpage source code or save some components such as the images included in the webpage (Aburrous et al., 2010 C). However, in our data set we have found that some legitimate websites try to block people from copying some elements included in their webpage in order to preserve their copyrights; therefore, they disable the right click. We have found this feature 40 times in our data set, 13 in the legitimate data set and the rest related to

phishing websites. This ration shows that this feature might not be significant in revealing phishing websites. Later in the experimental section we will assess whether this feature will be used in designing a new classifier or not. To extract this feature, we search for *event.button*==2 in the webpage source code.

4- **Using Pop-up Window**

It is unusual to find a legitimate website asking users to submit their information through a pop-up window. 127 webpages were encountered in phishing data set in which the users asked to submit their information through pop-up windows. Yet, this feature has been used in some legitimate websites to warn users about fraudulent activities or broadcast an announcement, though no personal information requested through pop-up windows. To extract this feature, we will check if the pop-up window (if exists) has any text fields such as username and password. If so, then the webpage is most likely supports phishing attempt.

5- **IFrame**

IFrame is an HTML tag used to display an additional webpage into the one that is currently shown. It may not be directly obvious that content within the webpage is from another source, since phishers depend on the design and colours of the webpage containing the *iframe*. In other words, phishers make use of the *frameBorder* attribute in order to hide the borders of the iframe. Hence, when the user browses a webpage, he might not recognise that another simultaneous page is loading in the *iframe* window. In some cases, legitimate websites may also use the *iframe* for different purposes such as showing an advertisement, but it is uncommon to find any fields asking for user's information within the *iframe*.

5.3.3.4. Domain based Features

1- **Age of Domain**

Most phishing websites live for a short period of time. In (Zhang et al., 2007) and (Xiang et al., 2011), the authors suggest that the webpage is legitimate if the domain age is 12 months or more.

2- **DNS Record**

For phishing websites, no record for the domain is found in the WHOIS database due to its short life (WHOIS, 2005) (Pan & Ding, 2006).

3- **Website Traffic**

This feature measures the popularity of the website by determining the number of visitors and the number of pages they visit. However, since phishing websites live for a short period of time they might not be recognized by the Alexa database (Alexa, 2011).

4- **PageRank**

PageRank is a value ranging from 1 to 10 (SEO, 2012). PageRank aims to measure how important a website is on the Internet. The greater the PageRank value, the more reliable the website (Garera et al., 2007). In our data sets, we have found that most phishing website either having no PageRank, or they have a low PageRank value due to short life span.

5- **Google Index**

This feature examines whether a website is indexed by Google or not. Any website indexed by Google is displayed on the search results (Google Search

Console, 2014). Usually, phishing websites are merely accessible for a short period, therefore, they are not found on the Google index (Zhang et al., 2007) (Xiang et al., 2011).

6- **Number of Links Pointing to the Webpage**

The number of links pointing to the webpage indicates its legitimacy level even if some links are from the same domain (Dean, 2013). In our data sets and due to their short life span, we have found that 2451 of phishing data set items have no links pointing to them. On the other hand, legitimate webpages have at least 2 external links pointing to them. This feature can be extracted from the Alexa database (Alexa, 2011).

7- **Statistical Reports based Feature**

Several parties such as PhishTank (PhishTank Stats, 2012), and StopBadware[8] (StopBadware, 2010) regularly formulate statistical reports on phishing websites. In our research, we used two forms of the Top 10 reports from PhishTank those are, monthly top 10 phishing domains and monthly top 10 phishing IPs published in the last three years. In addition, we used the top 50 IP addresses from StopBadware. If a URL has several features related to phishing and its host server has precedents of accommodating phishing websites then the website is most likely supports phishing attempt.

5.3.4. Collecting Training Data set

In order to facilitate collecting legitimate websites, we use the same resources commonly used in literature (Abdelhamid, 2015), (Mansour & Alshihri, 2015), (Basnet et al., 2012), (Alkhozae & Batarfi, 2011), (Aburrous et al., 2010 C),

[8] A non-profit anti-malware organization based in Cambridge, Massachusetts

(Miyamoto et al., 2008), (Abu-Nimeh et al., 2007), (Garera et al., 2007), (Pan & Ding, 2006), those are as follows:

1- Directory Mozilla (DMOZ) (Skrenta & Truel, 2011). DMOZ is the largest and most comprehensive human-edited directory of the Web.

2- One more source to collect legitimate websites is Alexa Top Sites (AWS, 2013). Alexa accompanies web browsers and works with them as they browse offering valuable information to Internet users about the websites' they are visiting and recommending related websites. The Alexa Top Sites offers access to lists of websites provided by Alexa Traffic Rank (Alexa, 2011). These lists are sorted from largest to smallest based on their rank which is computed by Alexa Traffic Rank.

In general, since our legitimate data set items are collected from legitimate sources we can certainly assume that our legitimate data set items are in fact confirmed real websites.

Phishing websites are collected from the PhishTank archive (PhishTank, 2011) and MillerSmiles archive (Bright, 2011). However, PhishTank is deemed the main source to collect phishing websites in this research. PhishTank depends on the so-called *wisdom of crowds*, i.e. considering the unanimous opinion of a group instead of a single expert to decide whether a website is in fact a phishing one or not. This ensures that incorrect reports will be ruled out. PhishTank provides downloadable databases available in multiple formats such as XML, CSV and PHP/JASON. We have considered downloading the database as a CSV format since it is much easier to deal with.

A data set consisting of 3869 instances has been collected. Such a data set comprises 1772 phishing websites and 2097 legitimate ones. Figure 5.12 shows the distribution of phishing and legitimate websites in our data set.

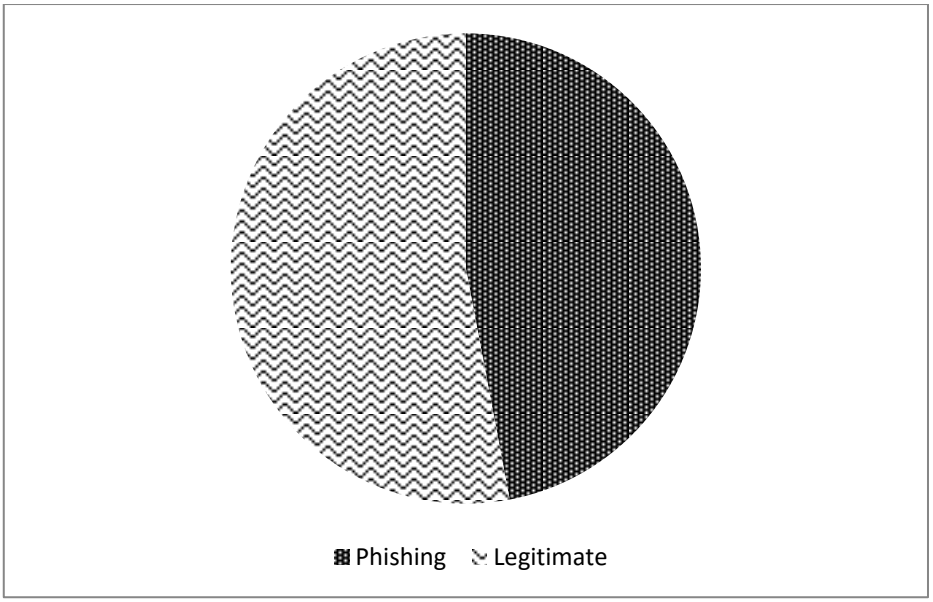


Figure 5.12 Distribution of phishing and legitimate websites in the training data set

5.3.5. Experimental Results

Three experiments have been done with the aim of evaluating the SSNN algorithm and compare the results with other DM classification algorithms. Information Gain (Shannon, 1948), Chi-Square (Greenwood & Nikulin, 1996) and Gain Ratio (Quinlan, 1993) have been used in experiments 1, 2 and 3 respectively. The results are shown in Tables 5.6 – 5.8.

Table 5.6 Experimental results when using Information Gain for features selection

		Number of selected features											
	Algorithm	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**	**Average**
Accuracy	**C4.5**	92.12	92.25	91.73	92.25	91.60	94.17	92.25	92.37	92.12	92.12	91.99	92.27
	BN	90.31	91.09	91.47	91.21	91.73	92.67	91.86	91.99	92.38	91.86	91.99	91.69
	LR	90.31	91.60	91.34	91.09	91.60	93.26	91.73	91.6	91.86	91.99	91.99	91.67
	FFNN	91.47	92.64	92.37	91.99	92.51	94.03	93.02	93.02	93.93	92.76	94.32	92.91
	SSNN-100	91.73	92.76	91.99	92.37	92.89	94.66	92.76	91.99	93.02	93.02	94.32	92.86
	SSNN-200	92.37	92.76	91.73	92.51	92.51	94.52	93.15	92.25	93.28	92.89	93.41	92.85
	SSNN-500	92.25	92.89	92.37	92.25	91.99	94.17	93.02	93.67	93.41	93.28	94.41	93.06
	SSNN-1000	91.99	92.63	91.99	92.25	92.25	94.39	93.15	93.15	93.41	93.54	94.06	92.98
	Average	91.57	92.33	91.87	91.99	92.14	93.98	92.62	92.51	92.93	92.68	93.31	92.54
F1-score	**C4.5**	91.50	91.60	91.10	92.40	90.90	93.40	91.50	91.60	91.40	91.40	91.20	91.64
	BN	89.10	90.00	90.50	90.30	90.80	91.70	91.00	91.10	91.60	91.00	91.10	90.75
	LR	89.10	90.70	90.40	90.10	90.70	92.40	90.90	90.80	91.10	91.30	91.30	90.80
	FFNN	90.50	91.70	91.50	91.00	91.70	93.40	92.20	92.30	93.40	92.00	93.80	92.14
	SSNN-100	90.90	91.90	90.90	91.40	92.20	94.10	91.90	91.10	92.30	92.50	93.80	92.09
	SSNN-200	91.70	91.90	90.70	91.60	91.70	93.90	92.40	91.40	92.60	92.20	92.80	92.08
	SSNN-500	91.60	92.00	91.60	91.40	91.20	93.50	92.30	93.10	93.40	92.50	92.70	92.30
	SSNN-1000	91.20	91.70	91.00	91.30	91.50	93.80	92.40	92.50	92.80	92.90	93.50	92.24
	Average	90.70	91.44	90.96	91.19	91.34	93.28	91.83	91.74	92.33	91.98	92.53	91.75
Recall	**C4.5**	92.70	92.10	92.10	89.90	91.00	91.20	90.40	89.90	90.70	90.70	90.70	91.04
	BN	85.70	87.60	88.50	88.50	88.80	89.60	89.60	89.60	90.20	89.30	89.60	88.82
	LR	85.70	89.30	88.80	88.50	89.30	90.60	90.20	90.20	90.70	91.00	91.00	89.57
	FFNN	88.80	88.80	89.30	87.90	90.20	93.60	89.90	90.70	92.70	91.00	93.80	90.61
	SSNN-100	89.90	88.80	87.40	88.20	90.70	93.70	89.30	89.00	90.40	93.30	94.10	90.44
	SSNN-200	91.90	88.80	87.60	88.80	90.20	93.60	90.20	89.30	91.00	91.60	92.10	90.46
	SSNN-500	92.40	88.80	89.90	89.60	90.20	92.40	91.00	92.40	94.40	90.20	91.00	91.12
	SSNN-1000	89.90	88.80	87.90	88.80	90.20	92.40	90.70	90.80	91.90	92.10	93.00	90.59
	Average	89.63	89.13	88.94	88.78	90.08	92.14	90.16	90.24	91.50	91.15	91.91	90.33
Pecision	**C4.5**	90.40	91.10	90.10	93.00	90.80	95.70	92.50	93.30	92.00	92.00	91.80	92.06
	BN	92.70	92.60	92.60	92.10	92.90	93.90	92.50	92.70	93.00	92.70	92.70	92.76
	LR	92.70	92.20	92.10	91.80	92.20	94.30	91.70	91.50	92.50	91.50	91.50	92.18
	FFNN	92.40	95.50	93.80	94.30	93.30	93.20	94.70	93.90	93.35	93.10	93.80	93.76
	SSNN-100	92.00	95.20	94.80	94.90	93.60	94.50	94.60	93.20	93.20	91.70	93.40	93.74
	SSNN-200	91.60	95.20	94.00	94.60	92.30	94.30	94.70	93.50	94.20	92.90	93.40	93.70
	SSNN-500	90.90	94.90	93.30	93.40	92.20	94.60	93.60	93.70	94.40	95.30	94.50	93.71
	SSNN-1000	92.50	94.90	94.30	93.60	92.40	94.20	94.20	93.70	93.70	93.70	94.00	93.75
	Average	91.90	93.95	93.13	93.46	92.46	94.34	93.56	93.19	93.29	92.86	93.14	93.21

Table 5.7 Experimental results when using Chi-Square for features selection

		Number of selected features											
	Algorithm	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**	**Average**
Accuracy	**C4.5**	92.12	92.25	91.73	92.25	91.60	91.73	92.25	92.64	92.12	92.12	91.99	92.07
	BN	90.31	91.09	91.47	91.21	91.73	91.60	91.86	92.12	92.38	91.86	91.99	91.60
	LR	90.31	91.60	91.34	91.09	91.60	91.60	91.73	92.25	91.86	91.99	91.99	91.58
	FFNN	89.02	93.02	92.37	92.51	91.99	92.76	93.15	93.93	93.54	93.67	93.02	92.63
	SSNN-100	91.99	92.76	92.37	91.73	92.64	92.38	93.28	93.28	93.67	93.67	93.66	92.86
	SSNN-200	91.99	92.64	92.37	92.37	92.76	92.25	93.67	93.92	93.67	93.67	93.02	92.94
	SSNN-500	92.62	92.64	92.37	92.51	91.86	92.40	93.90	93.15	93.67	93.67	93.54	92.94
	SSNN-1000	92.24	92.76	92.51	93.02	92.25	92.25	93.67	93.28	94.06	92.57	93.02	92.88
	Average	91.33	92.35	92.07	92.09	92.05	92.12	92.94	93.07	93.12	92.90	92.78	92.44
F1-score	**C4.5**	91.50	91.60	91.10	91.40	90.90	90.90	91.50	92.00	91.40	91.00	91.20	91.32
	BN	89.10	90.00	90.50	90.30	90.80	90.70	91.00	91.30	91.60	91.00	91.10	90.67
	LR	89.10	90.70	90.40	90.10	90.70	90.70	90.90	91.50	91.10	91.30	91.30	90.71
	FFNN	88.80	92.20	91.50	91.60	91.20	91.90	92.50	93.30	92.80	92.90	92.20	91.90
	SSNN-100	91.40	91.90	91.60	90.90	91.90	91.50	92.60	92.60	93.00	93.00	93.00	92.13
	SSNN-200	91.40	91.70	91.60	91.60	92.00	91.30	93.00	92.30	92.90	93.00	92.20	92.09
	SSNN-500	91.70	91.70	91.40	91.70	91.10	91.50	93.20	92.40	93.00	93.00	92.80	92.14
	SSNN-1000	91.60	91.90	91.70	92.30	91.50	91.30	93.00	92.60	93.30	92.00	92.20	92.13
	Average	90.58	91.46	91.23	91.24	91.26	91.23	92.21	92.25	92.39	92.15	92.00	91.64
Recall	**C4.5**	92.70	92.10	92.10	89.90	91.00	89.30	90.40	91.90	90.70	89.30	90.70	90.92
	BN	85.70	87.60	88.50	88.50	88.80	89.00	89.60	89.90	90.20	89.30	89.60	88.79
	LR	85.70	89.30	88.80	88.50	89.30	89.30	90.20	91.00	90.70	91.00	91.00	89.53
	FFNN	89.00	89.30	89.30	89.00	90.20	89.00	91.60	91.30	91.00	90.70	89.90	90.03
	SSNN-100	92.40	89.00	90.40	89.30	90.40	89.30	91.00	91.90	91.10	91.10	91.10	90.64
	SSNN-200	92.40	88.80	89.90	89.90	91.00	88.80	92.10	91.60	90.70	92.30	89.90	90.67
	SSNN-500	91.90	89.00	89.80	89.60	90.40	89.80	91.90	91.30	91.90	91.30	90.70	90.69
	SSNN-1000	92.40	89.00	89.90	91.00	90.70	88.50	91.90	91.30	91.60	89.90	89.90	90.55
	Average	90.28	89.26	89.84	89.46	90.23	89.13	91.09	91.28	90.99	90.61	90.35	90.23
Pecision	**C4.5**	90.40	91.10	90.10	93.00	90.80	92.40	92.50	92.10	92.00	92.00	91.80	91.65
	BN	92.70	92.60	92.60	92.10	92.90	92.40	92.50	92.80	93.00	92.70	92.70	92.64
	LR	92.70	92.20	92.10	91.80	92.20	92.20	91.70	92.00	91.50	91.50	91.50	91.95
	FFNN	90.30	95.20	93.80	94.30	92.20	94.90	93.40	94.30	94.70	93.10	94.70	93.72
	SSNN-100	90.40	94.90	92.80	92.40	93.30	93.80	94.20	93.40	94.00	94.00	94.00	93.38
	SSNN-200	90.40	94.90	93.30	93.30	93.10	94.00	94.00	95.00	95.30	93.80	93.70	93.71
	SSNN-500	91.60	93.90	93.90	93.40	91.60	94.40	94.40	93.90	94.20	94.80	94.50	93.69
	SSNN-1000	90.90	94.90	93.60	93.60	92.30	94.30	94.20	93.90	95.30	93.90	93.70	93.69
	Average	91.18	93.71	92.78	92.99	92.30	93.55	93.36	93.43	93.75	93.23	93.33	93.05

Table 5.8 Experimental results when using Gain Ratio for features selection

		Number of selected features											
	Algorithm	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**	**Average**
Accuracy	**C4.5**	92.12	92.12	92.12	91.60	91.60	91.99	92.25	92.12	92.24	92.24	92.25	92.06
	BN	90.31	91.21	91.09	91.47	91.73	92.12	91.86	91.86	91.99	91.86	92.25	91.61
	LR	90.31	91.86	91.86	92.12	91.60	91.86	91.73	91.86	91.86	91.86	91.86	91.71
	FFNN	91.47	91.99	89.92	91.09	92.51	92.76	92.76	92.64	92.64	92.89	93.54	92.20
	SSNN-100	91.99	92.24	91.37	91.09	93.02	92.89	93.54	92.41	93.15	93.02	92.89	92.51
	SSNN-200	91.99	92.12	90.95	90.70	92.38	93.54	93.02	92.41	93.67	93.67	93.80	92.57
	SSNN-500	92.25	92.37	90.05	91.09	91.99	93.15	93.28	93.28	93.67	93.67	93.67	92.59
	SSNN-1000	92.25	92.37	90.17	91.09	92.63	92.78	93.28	93.28	93.67	93.28	93.67	92.59
	Average	91.59	91.99	90.94	91.31	92.18	92.64	92.63	92.37	92.75	92.81	92.89	92.23
F1-score	**C4.5**	91.50	91.50	91.50	91.00	90.90	91.30	91.50	91.30	91.40	91.40	91.50	91.35
	BN	89.10	90.20	90.10	90.50	90.80	91.30	91.00	91.00	91.10	91.00	91.40	90.68
	LR	89.10	91.10	91.10	91.30	90.70	91.10	90.90	91.10	91.10	91.10	91.10	90.88
	FFNN	90.50	91.10	88.90	90.20	91.70	92.10	91.90	91.80	91.90	92.20	92.90	91.38
	SSNN-100	91.40	91.60	91.00	90.20	92.30	92.30	92.90	91.70	92.50	92.40	91.89	91.84
	SSNN-200	91.40	91.50	90.20	89.70	91.60	92.90	92.30	91.70	93.00	93.10	92.80	91.84
	SSNN-500	91.60	91.70	89.00	90.20	91.10	92.50	92.60	92.60	93.00	93.00	93.00	91.85
	SSNN-1000	91.60	91.70	89.10	90.20	91.80	92.20	92.60	92.60	92.90	92.60	92.90	91.84
	Average	90.66	91.24	90.11	90.44	91.36	91.96	91.87	91.60	92.00	92.10	92.08	91.46
Recall	**C4.5**	92.40	92.10	90.70	88.80	90.70	92.70	91.30	90.60	91.30	91.10	92.20	91.26
	BN	85.70	88.20	88.50	88.50	88.80	89.60	89.60	89.60	89.60	89.60	89.60	88.85
	LR	85.70	90.20	90.20	89.90	89.30	90.40	90.20	90.20	90.40	90.40	90.20	89.74
	FFNN	88.80	88.80	87.40	88.80	90.20	92.90	89.30	89.60	91.00	91.00	91.60	89.95
	SSNN-100	92.70	92.70	92.70	92.10	91.00	91.00	90.40	90.40	89.90	89.90	90.20	91.18
	SSNN-200	92.40	91.90	90.40	87.90	90.70	92.10	91.00	90.60	91.30	92.40	92.10	91.16
	SSNN-500	92.40	91.90	87.60	88.80	90.20	91.90	90.70	91.00	91.30	91.90	91.30	90.82
	SSNN-1000	92.40	91.90	87.60	88.80	89.90	91.60	90.70	91.00	90.40	91.00	91.60	90.63
	Average	90.01	90.96	89.39	89.26	90.10	91.53	90.36	90.29	90.65	90.91	91.10	90.45
Pecision	**C4.5**	90.40	90.40	90.40	89.90	90.80	91.50	92.50	92.30	93.00	93.00	92.80	91.55
	BN	92.70	92.40	91.80	92.60	92.90	93.00	92.50	92.50	92.70	92.50	93.30	92.63
	LR	92.70	92.00	92.00	92.80	92.20	91.70	91.70	92.50	91.70	91.70	92.00	92.09
	FFNN	92.40	93.50	90.40	91.60	93.30	92.40	94.60	94.10	92.80	93.40	94.20	92.97
	SSNN-100	90.40	91.10	91.20	91.60	93.90	91.90	94.50	92.90	93.70	92.70	93.40	92.48
	SSNN-200	90.40	91.10	89.90	91.50	92.60	93.70	93.60	92.90	94.80	93.70	94.00	92.56
	SSNN-500	90.90	91.60	90.40	91.60	93.00	93.10	94.40	94.20	94.80	94.20	94.80	93.00
	SSNN-1000	90.90	91.60	90.70	91.60	93.80	92.90	94.40	94.20	95.50	94.20	93.05	92.99
	Average	91.41	91.73	90.85	91.66	92.81	92.53	93.40	93.06	93.63	93.18	93.44	92.53

From the results demonstrated in Tables 5.6, 5.7 and 5.8, we can see that the SSNN algorithm outperforms the considered classification algorithms in most cases, particularly when the epoch size is set to 500. These results are consistent with the results achieved in sections 5.2.4 and 3-. For instance, the average F1-

score produced from the SSNN when using the Information Gain is higher than that produced from C4.5, BN, LR, and FFNN with margins of 0.66, 1.55%, 1.50%, and 0.16% respectively when the epoch size is set to 500. In addition, the average F1-score produced from the SSNN when using the Chi-Square is also beats C4.5, BN, LR, and FFNN with margins of 0.82%, 1.46%, 1.43%, and 0.24% respectively when the epoch size is set to 500. Again, when using the same epoch size, the average F1-score produced from the SSNN algorithm when using the Gain Ration outperforms C4.5, BN, LR, and FFNN with margins of 0.50%, 1.16%, 0.96%, and 0.46% respectively. Overall, the high F1-score yielded from the SSNN reflects that the algorithm is able to derive classifiers that produce good FP and FN rates. That can be attributed due to the well-structured NN classifiers derived from the SSNN algorithm as a result to the good training procedure employed in the algorithm. In general, the F1-score produced when using different feature selection methods reflects that the NN based algorithms derive better classifiers than other considered classification algorithms when applied to phishing datasets in the sense that the second best result achieved in all experiments was from the FFNN. However, the highest F1-score produced from the SSNN was when using the Information Gain for feature selection at 92.30%. This value is higher than the values produced from Chi-Square and Gain Ration with margins of 0.16% and 0.45% respectively.

In terms of average accuracy, and when the epoch size is set to 500, the SSNN outperforms C4.5, BN, LR, and FFNN with margins of 0.79%, 1.38%, 1.39%, and 0.15% respectively when using the Information Gain. In addition, the average accuracy produced from the SSNN algorithm when using the Chi-square outperforms C4.5, BN, LR, and FFNN with margins of 0.87%, 1.34%, 1.36%, and 0.30% respectively. Further, the SSNN algorithm beats C4.5, BN, LR, and FFNN with margins of 0.53%, 0.97%, 0.88% and 0.39% respectively when using the Gain Ration. Overall, the high accuracies produced from the SSNN when using

different feature selection methods are good sign that the training procedure in the SSNN algorithm is able to produce well-structured NN classifiers. Yet, the highest average accuracy produced from the SSNN was when the Information Gain has been used for feature selection at 93.06%. This value bypasses the results achieved from Chi-Square and Gain Ration with margins of 0.12% and 0.47% respectively.

In terms of average Recall, we find that the SSNN algorithm has been defeated two times from the C4.5 when using Chi-Square and Gain Ration with margins of 0.23% and 0.45% respectively when the epoch size is set to 500. However, when using the Information Gain, we find that the SSNN outperforms the C4.5 with a margin of 0.08%. This difference is relatively small. However, a good classification model is the model that is able to maximize both Precision and Recall simultaneously. Yet, from the results, we find that although the average Recall produced from C4.5 beats the SSNN algorithm when using Chi-Square and Gain Ration, the SSNN algorithm outperforms the C4.5 in terms of average Precision with 2.04% and 1.45% when using Chi-Square and Gain Ration respectively when the epoch size is set to 500. Such results confirm that the SSNN algorithm is able to derive classifiers that show a moderately good performance on both Precision and Recall. The same scenario is also happens with FFNN, since the FFNN produced higher precision than the SSNN with margins of 0.03% when using Chi-Square when the epoch size is set to 500. Yet, when using the same epoch size, the SSNN produced higher Recalls than the FFNN with margins of 0.66% and 0.87% when using Chi-Square and Gain Ration respectively.

Overall, the training procedure utilised when deriving NN classifiers using the SSNN algorithm has proven to be effective in creating well-structured models in terms of number of hidden neurons and weights space.

5.3.6. Comparison across Different Feature Selection Methods

The experimental results revealed that the Information Gain is more effective in improving the performance of the SSNN and other considered classification algorithms when applied to phishing websites data set. For instance, the average accuracy produced from the SSNN and other compared classification algorithms when using the Information Gain is higher than the average accuracy produced from the Chi-Square and Gain Ration with margins of 0.10% and 0.31% respectively. In addition, the average F1-Score produced from the SSNN and other compared classification algorithms when using the Information Gain is 91.75%. This value is higher than that produced from the Chi-Square and Gain Ration with margins of 0.11% and 0.29%. More details can be viewed in Tables 5.6 – 5.8. These results reveal that the Information Gain is more effective in improving the performance of the SSNN and other considered classification algorithms when applied to phishing websites classification problem. These results are consistent with the results obtained in (Akinyelu & Adewumi, 2014), (Ramanathan & Wechsler, 2012), (Al-Momani et al., 2011), (Whittaker et al., 2010), (Ma et al., 2009), and (Chandrasekaran et al., 2006).

5.4. Chapter Summary

In this chapter, several experiments have been accomplished to assess the performance of the classifiers that are created using the SSNN algorithm. Two sets of experiments have been conducted. In the first set, the SSNN has been assessed against several DM classification algorithms on a number of binary data sets from UCI repository. The experimental data sets vary in size, i.e. number of instances, and number of attributes. The experimental results show that the NN based algorithms, i.e. FFNN and SSNN produce classifiers better

than C4.5 and BN in terms of error-rate. From the results, we find that the classifiers produced from the SSNN when epoch size is set to 500 have the least error-rate on average. In terms of RAR, we find that SSNN produces higher RAR in most cases. However, the average time needed for the SSNN to create classifiers is higher than the average time needed for other contrasted algorithms. That could be attributed to the fact that the SSNN performs several sub-trainings in each training session before producing the final classifier. In general, this set of experiments show that the SSNN algorithm is able to produce good classifiers with good generalisation ability.

The second set of experiments evaluates the applicability of the SSNN on phishing websites data set. Three feature selection methods have been used in order to evaluate these methods and their effect on the performance of the SSNN and other considered classification algorithms. The results show that the SSNN algorithm outperformed the considered classification algorithms in most cases. The classifiers produced from the SSNN have been shown to produce a moderately good performance on both Precision and Recall. However, the experimental results revealed that the Information Gain is more effective than other feature selection methods in improving the performance of the SSNN and other considered classification algorithms.

In the next chapter, the SSNN algorithm will be used to create the classifiers that are added to the ESSNN.

CHAPTER 6

An Application of ESSNN to Phishing Websites

6.1. Introduction

In this chapter, the ESSNN is applied to a major web security problem where a virtual concept drift might occur, that is the problem of classifying phishing websites into genuine or phishing. The aims of applying the ESSNN to phishing websites are: firstly, empirically evaluate the applicability of ESSNN to phishing websites and compare its results against other approaches. Secondly, assess if the ESSNN can handle the virtual concept drift and offer a balance between stability and plasticity to cope with the dynamic nature of the phishing websites classification problem.

Two sets of experimental evaluation are conducted. In the first set, we will compare the performance of ESSNN against several offline DM algorithms. As shown in sections 3.8 and 3.9.7, most DM based phishing websites classification techniques are devoted to an offline learning strategy according to which as soon as the anti-phishing model is created it can obtain no further knowledge. Therefore, the first set of experiments aims to compare the results when we treat phishing websites as a dynamic problem, with the results when we treat it as a static problem.

In the second set of experiments, we will compare the performance of ESSNN against two different single classifier based stream mining algorithms (window based algorithms). As shown in sections 3.6 and 3.7 phishing websites is a dynamic classification problem where a virtual concept drift might occur. However, as we have discussed in section 2.8 two possible approaches can be used to handle the concept drift and offer a balance between stability and plasticity those are: a single classifier based approach and an ensemble based approach. Yet, the concept drift that occurs in phishing websites is a cyclical concept drift, therefore, the main aim of this set of experiments is to assess the performance of ESSNN against two single classifier based stream mining algorithms in handling the cyclical concept drift that characterises the phishing websites. When a classifier is created using single classifier based algorithms, the window size is the most important factor that affects the classifier performance. Therefore, several window size values are used to derive several classifiers.

We will use the same validation technique and evaluation metrics as described in sections 5.2.3, and 5.3.2 to assess the experimental results. Several DM algorithms have been used; these algorithms are reviewed in section 6.2. Three data sets have been collected in different time span. These data sets are described in section 6.3. After conducting several experiments in sections 6.4 and 6.5, the results are discussed in section 6.6.

6.2. Algorithms used for Comparison

Two sets of experiments will be conducted. In the first set, five different algorithms have been selected, those are, Bayesian Network (Friedman et al., 1997), Decision Trees (Quinlan, 1998), Support Vector Machine (Cortes & Vapnik, 1995), Logistic Regression (Witten et al., 2011), and RIPPER (Cohen, 1995). These algorithms have been briefly discussed in section Appendix A.

The selection of these algorithms is because they are commonly used in the domain of phishing classification (Abu-Nimeh et al., 2007), (Zhang et al., 2007), (Fette et al., 2007), (Miyamoto et al., 2008), (Basnet et al., 2008), (Aburrous et al., 2010 C), and (Abdelhamid et al., 2014). The experiments on these algorithms are carried out using the WEKA platform (Hall et al., 2011).

In the second set of experiments, we will compare the performance of ESSNN against two single classifier based stream mining algorithms in terms of handling the virtual concept drift that occurring in phishing websites. The VFDT (Domingos & Hulten, 2000) has been used and coupled with HAT algorithm (Bifet & Gavaldà, 2009). These algorithms have been selected because the hoeffding tree based algorithms represent the current state-of-the-art in single classifier based stream mining algorithms (Gama et al., 2014). However, each algorithm applies different forgetting procedure. For instance, the VFDT applies a static forgetting procedure, i.e. the examples that currently exist in the window will be deleted as soon as a new data set batch arrives. Yet, the HAT algorithm applies a dynamic forgetting procedure, in the sense that each node in the tree is able to determine which of the pervious examples should be maintained and used in the next training session. The experiments on these algorithms are carried out using the Massive Online Analysis platform (MOA) (Bifet et al., 2010). MOA is an open source framework for dealing with evolving data streams. MOA is written in Java and contains several online learning algorithms. ESSNN has been implemented in Java. All experiments were conducted in a system with CPU Pentium Intel® CoreTM i5-2430M @ 2.40 GHz, RAM 4.00 GB. The platform is Windows 7 64-bit Operating System.

6.3. Training Data sets

We have used the same data sets collection strategy as shown in section 5.3.4. Three data sets have been collected in different time span. Several PHP and JavaScript codes are written to extract the phishing websites features.

The first data set is collected from October-28-2013 until January-8-2014. The data set consisted of 2456 instances, 1094 of them were legitimate websites and 1362 were phishing ones.

The second data set is gathered from May-13-2014 until September-24-2014. This data set contained 1355 phishing websites and 1686 legitimate ones. The complete data set included 3041 instances.

The last data set comprised of 1264 legitimate websites and 1624 phishing ones. The total is 2887 instances. This data set was collected from December-8-2014 until April-20-2015.

Table 6.1 shows a detailed description of the collected data sets. Figure 6.1 shows the distribution of phishing and legitimate websites in our data sets.

Table 6.1 Description of training data sets

Data Set	Number of Phishing instances	Number of Legitimate Instances	Total Number of Instances
1	1362	1094	2456
2	1355	1686	3041
3	1624	1263	2887

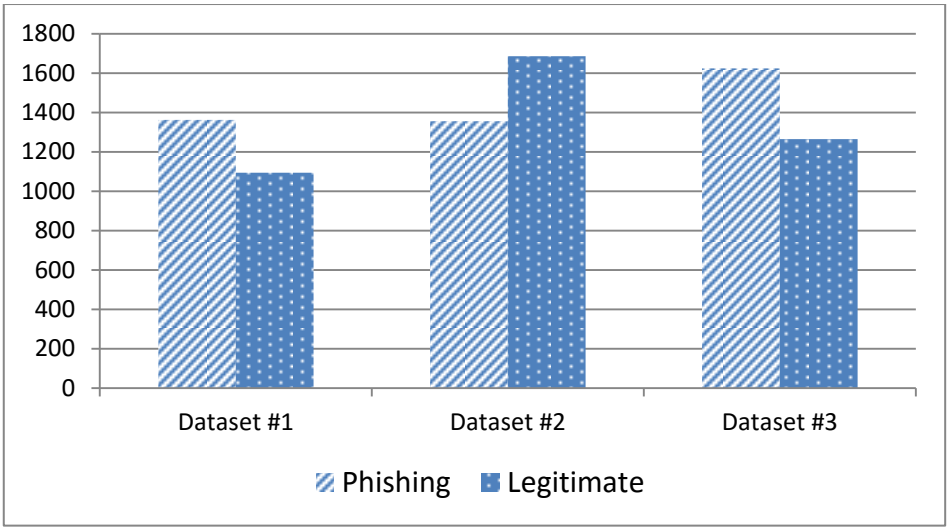


Figure 6.1 Distribution of phishing and legitimate websites

6.4. ESSNN vs Other Data Mining Algorithms

6.4.1. Experimental Strategy

Three experiments will be conducted. Data set #1, Data set #2 and Data set #3 described in section 6.3 are used in experiments 1, 2, and 3 respectively. In each experiment, we will start by selecting the most effective set of features. The experimental results obtained in section 5.3 showed that the results when using the Information Gain were the highest among other feature selection methods; therefore, the Information Gain will be used for selecting the most effective set of feature in each experiment. Information Gain is a well-known method to evaluate features for significance for the classification task (Bramer, 2013), (Dash & Liu, 1997), (Yu & Liu, 2004) and is used in many classification domains, e.g. (Quinlan & Kaufmann, 1993), (Lee & Xiang, 2001), (Khemphila & Boonjing, 2011), (Yang & Pedersen, 1997), (Zhang et al., 2003) and (Basnet et al., 2012). However, since there is no agreement on a specific ranking threshold for picking the most effective set of features, several threshold values will be used. The threshold values utilised in our experiments are 1%, 10%, 20% and 30%.

The features that pass the threshold value will produce a new data set (minimal data set). Consequently, several minimal data sets will be created.

The set of features in each minimal data set are compared with the features used in creating the classifiers currently existing in the ESSNN. If the features are different, then a virtual concept drift occurs. Therefore, the minimal data set is used to create a new classifier.

We will use the default parameter settings of WEKA (Hall et al., 2011) when creating the classifiers from the considered DM algorithms. However, when creating the SSNN classifiers, two parameters should be specified, i.e. the number of epochs and maximum number of possible hidden neurons. Following some recent studies (Kesari et al., 2014), (Madhusmita et al., 2012), (Ganatra et al., 2011), (Insung & Wang, 2007), (Hossein et al., 2003), (Shamik et al., 2011), (Paulin & Santhakumaran, 2011), (Amato et al., 2013) we set the maximum number of hidden neurons to 15. However, we have used the same epoch size as in FFNN; therefore, we set the epoch size to 500 in the following set of experiments. This epoch size value has also been shown to produce the best results as revealed in chapter 5. As soon as a new classifier is created from the SSNN algorithm, it will be temporarily added to the ESSNN to assess the improvement the classifier adds to the ESSNN performance. The classifier that best improves the ESSNN performance will be added permanently to the ESSNN. However, deriving a new classifier from other DM algorithms means that the old classifier is removed. The performance of the ESSNN in each experiment as well as other considered DM algorithms will be assessed on a testing data set consisting of the testing data set achieved from the data set used in the experiment and the historical testing data sets (except in the first experiment where there are no historical data sets). The main aim of merging the most recent testing data set with all historical ones is to assess the

performance of ESSNN and other considered DM algorithms in case a cyclical concept drift occurs. Finally, the set of examples that are misclassified will be kept and used when creating a new classifier.

6.4.2. Results from the First Data set

Twelve features have been selected in this experiment when the threshold value is set to 1% (minimum considered threshold), such features are shown in Table 6.2.

Table 6.2 Selected features when threshold=1% (First Data set)

Rank	IG value	Feature
1	0.4938	HTTPS
2	0.4920	URL of Anchor
3	0.2499	Adding Prefix or Suffix to domain
4	0.2400	Web traffic
5	0.1169	PageRank
6	0.0921	Age of Domain
7	0.0769	Sub Domain and Multi Sub Domains
8	0.0548	Links in <Meta>, <Script> and <Link> tags
9	0.0537	Using Free Domain Registration
10	0.0514	Request URL
11	0.0245	DNS Record
12	0.0165	Google Index

At the end of this experiment, the ESSNN will consist of one classifier. Figure 6.2 and

Table 6.3 show the performance of the ESSNN and other considered algorithms on several threshold values.

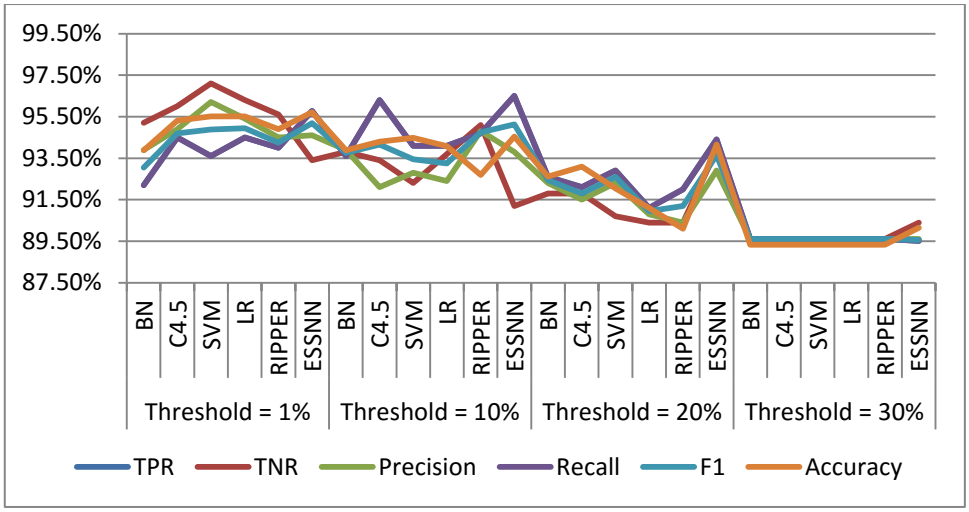


Figure 6.2 Performance of ESSNN and other DM algorithms (First Data set)

Table 6.3 Performance of ESSNN and other DM Algorithms (First Data set)

Threshold	Algorithm	#Features	TPR	TNR	Precision	Recall	F1	Accuracy
Threshold = 1%	BN	12	92.20%	95.20%	93.90%	92.20%	93.04%	93.88%
	C4.5	12	94.50%	96.00%	94.90%	94.50%	94.70%	95.31%
	SVM	12	93.60%	97.10%	96.20%	93.60%	94.88%	95.51%
	LR	12	94.50%	96.30%	95.40%	94.50%	94.95%	95.51%
	RIPPER	12	94.00%	95.60%	94.50%	94.00%	94.25%	94.89%
	ESSNN	12	95.78%	93.40%	94.60%	95.78%	95.19%	95.71%
Threshold = 10%	BN	5	93.60%	93.80%	93.90%	93.60%	93.75%	93.88%
	C4.5	5	96.30%	93.40%	92.10%	96.30%	94.15%	94.29%
	SVM	5	94.10%	92.30%	92.80%	94.10%	93.45%	94.48%
	LR	5	94.10%	93.70%	92.40%	94.10%	93.24%	94.08%
	RIPPER	5	94.70%	95.10%	94.80%	94.70%	94.75%	92.69%
	ESSNN	5	96.50%	91.20%	93.80%	96.50%	95.13%	94.54%
Threshold = 20%	BN	4	92.60%	91.80%	92.30%	92.60%	92.45%	92.62%
	C4.5	4	92.10%	91.80%	91.50%	92.10%	91.80%	93.09%
	SVM	4	92.90%	90.70%	92.30%	92.90%	92.60%	92.04%
	LR	4	91.10%	90.40%	90.80%	91.10%	90.95%	91.10%
	RIPPER	4	92.00%	90.40%	90.40%	92.00%	91.19%	90.10%
	ESSNN	4	94.40%	93.90%	92.90%	94.40%	93.64%	94.15%
Threshold = 30%	BN	2	89.60%	89.60%	89.60%	89.60%	89.60%	89.33%
	C4.5	2	89.60%	89.60%	89.60%	89.60%	89.60%	89.33%
	SVM	2	89.60%	89.60%	89.60%	89.60%	89.60%	89.33%
	LR	2	89.60%	89.60%	89.60%	89.60%	89.60%	89.33%
	RIPPER	2	89.60%	89.60%	89.60%	89.60%	89.60%	89.33%
	ESSNN	2	89.51%	90.40%	89.60%	89.51%	89.55%	90.14%

From the results, we can see that the ESSNN outperforms BN, C4.5, SVM, LR, and RIPPER in terms of F1-score with a margin of 2.15%, 0.49%, 0.31%, 0.24%, and 0.94% respectively when the threshold value is set to 1%. That means that the ESSNN shows a moderately good performance on both P and R. At the

same threshold value, we find that the ESSNN outperforms BN, C4.5, SVM, LR, and RIPPER with 1.83%, 0.40%, 0.20%, 0.20%, and 0.82% in terms of produced accuracy. That can be attributed to the training procedure of SSNN algorithm that is able to produce a well-structured NN classifier in terms of number of hidden neurons and weights space since SSNN algorithm leaves the network expansion as a last option and keeps pushing on the learning rate by adjusting the desired error rate before adding a new hidden neuron. SVM produces the highest TNR among others when threshold value is set to 1%. Yet, SVM produces the second lowest TPR after the BN. That means that the SVM does not produce a moderate balance between Precision and Recall as we can see in

Table 6.3 since it favours Precision over Recall. Overall, the performance of the ESSNN and other considered DM algorithms when threshold value is set to 1% is acceptable, and that indicates features goodness in predicting the website class. However, the performance of the ESSNN and other algorithms decays when the threshold value is set to 30%. That can be attributed to the fact that there is not enough information to learn the classifiers because only two features are selected at this threshold these are HTTPS and URL of anchor.

6.4.3. Results from the Secnod Data set

After applying the IG on the second data set, we find that some features drop from the list that has been used to create the classifier that currently exists in the ESSNN, i.e. the classifier created using the first data set. These features are age of domain, DNS record and Google index. On the other hand, several features are added to the list, those are using IP address, suspicious action upon submitted information, .htaccess redirect, and statistical based features. Hence, a concept drift occurs. Table 6.4 shows the selected features when threshold value is set to 1%.

Table 6.4 Selected features when threshold=1% (Second Data set)

Rank	IG value	Feature
1	0.5421	HTTPS
2	0.4735	URL of Anchor
3	0.2770	Web traffic
4	0.2635	Adding Prefix or Suffix to domain
5	0.1506	Sub Domain and Multi Sub Domains
6	0.1342	PageRank
7	0.0953	Suspicious Action upon Submitted Information
8	0.0939	Links in <Meta>, <Script> and <Link> tags
9	0.0675	Using Free Domain Registration
10	0.0554	Request URL
11	0.0307	Using IP Address
12	0.0220	*.htaccess* Redirect
13	0.0195	Statistical Report-based features

To assess the effect of these changes on the ESSNN performance, we consider the second data set as a testing data set and pass it to the ESSNN. The results show that the overall accuracy drops from 95.71% to 92.28% due to the fact that a concept drift has occurred. Therefore, we will create a new classifier using the SSNN algorithm and add it to the ESSNN.

Since we are using four threshold values for features selection, four classifiers will be created. The performance of the ESSNN and other considered DM algorithms on different threshold values are depicted in Figure 6.3 and Table 6.5. The testing data set used to assess the performance is a combination of the testing data sets from the first and the second data sets. By merging these two testing data sets, we aim to assess the performance of the ESSNN and other considered algorithms if a cyclical concept drift occurs.

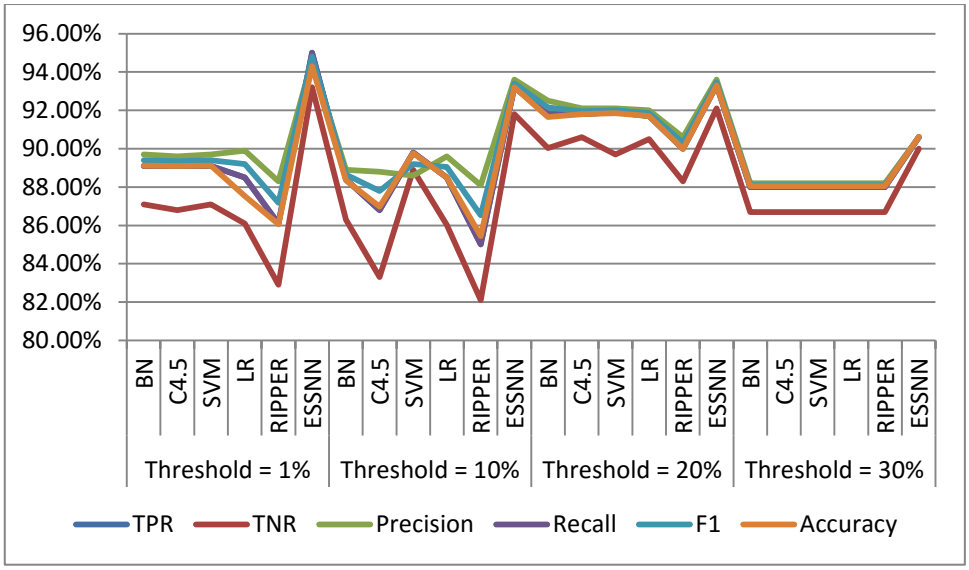


Figure 6.3 Performance of SSNN and other DM algorithms (Second Data set)

Table 6.5 Performance of ESSNN and other DM algorithms (Second Data set)

Threshold	Algorithm	#Features	TPR	TNR	Precision	Recall	F1	Accuracy
Threshold = 1%	BN	13	89.10%	87.10%	89.70%	89.10%	89.40%	89.11%
	C4.5	13	89.10%	86.80%	89.60%	89.10%	89.35%	89.10%
	SVM	13	89.10%	87.10%	89.70%	89.10%	89.40%	89.11%
	LR	13	88.50%	86.10%	89.90%	88.50%	89.19%	87.53%
	RIPPER	13	86.10%	82.90%	88.30%	86.10%	87.19%	86.06%
	ESSNN	13	95.00%	93.20%	94.60%	95.00%	94.80%	94.31%
Threshold = 10%	BN	6	88.40%	86.30%	88.90%	88.40%	88.65%	88.35%
	C4.5	6	86.80%	83.30%	88.80%	86.80%	87.79%	86.97%
	SVM	6	89.80%	88.90%	88.60%	89.80%	89.20%	89.77%
	LR	6	88.50%	86.03%	89.60%	88.50%	89.05%	88.50%
	RIPPER	6	85.00%	82.10%	88.10%	85.00%	86.52%	85.45%
	ESSNN	6	93.20%	91.80%	93.60%	93.20%	93.40%	93.21%
Threshold = 20%	BN	4	91.80%	90.04%	92.50%	91.80%	92.15%	91.66%
	C4.5	4	91.80%	90.60%	92.10%	91.80%	91.95%	91.81%
	SVM	4	91.90%	89.70%	92.10%	91.90%	92.00%	91.86%
	LR	4	91.71%	90.50%	92.00%	91.71%	91.85%	91.71%
	RIPPER	4	90.00%	88.30%	90.60%	90.00%	90.30%	89.98%
	ESSNN	4	93.30%	92.10%	93.60%	93.30%	93.45%	93.31%
Threshold = 30%	BN	2	88.00%	86.70%	88.20%	88.00%	88.10%	%88.04
	C4.5	2	88.00%	86.70%	88.20%	88.00%	88.10%	%88.04
	SVM	2	88.00%	86.70%	88.20%	88.00%	88.10%	%88.04
	LR	2	88.00%	86.70%	88.20%	88.00%	88.10%	%88.04
	RIPPER	2	88.00%	86.70%	88.20%	88.00%	88.10%	%88.04
	ESSNN	2	90.60%	90.00%	90.60%	90.60%	90.60%	%90.61

The ESSNN achieves the best results when the classifier created at threshold=1% is added to the ensemble, hence, this classifier will be added to the ESSNN and the others will be ignored. At that threshold value, the ESSNN achieves 5.05%, 5.10%, 5.05%, 5.26%, and 7.26% higher F1-score than BN, C4.5, SVM, LR, and RIPPER respectively. In addition, the ESSNN outperforms BN, C4.5, SVM, LR, and RIPPER with 5.20%, 5.21%, 5.20%, 6.78%, and 8.25% in terms of overall accuracy.

These results show that the ESSNN can effectively handle the virtual concept drift because the ESSNN contains two classifiers each of which is considered an expert on different features space. On the contrary, the considered DM algorithms are ineffective in handling the virtual concept drift that occurs in phishing websites because of the catastrophic forgetting dilemma. Another reason that might also contribute to the good results achieved from the ESSNN is that the ESSNN supports reinforcement learning (Cohen & Maimon, 2005)

because the set of misclassified examples in the first data set has been merged with the second data set and used in learning a new classifier. One more reason for the good results produced from the ESSNN is that it uses an effective technique in producing the final decision, i.e. weighted voting technique. However, when the threshold value is set to 20% and 30%, we notice that the performance of the ESSNN and other considered algorithms are very close to the results achieved from the first data set on the same threshold values. That is because the set of features that constitutes the testing data set (which is a combination of testing data set from the first and the second data set) has the same significance degree in both data sets. These results support our suggestion about not building a new classifier if the set of features has been used in creating a previous classifier since this will result in deriving redundant classifiers.

6.4.4. Results from the Third Data set

The set of features achieved from the IG in this data set when threshold value is set to 1% are shown in Table 6.6.

Table 6.6 Selected Features (Third Data set)

Rank	IG value	Feature
1	0.4939	HTTPS
2	0.4871	URL of Anchor
3	0.2630	Sub Domain and Multi Sub Domains
4	0.2293	Adding Prefix or Suffix to domain
5	0.1788	Age of Domain
6	0.1231	Web traffic
7	0.0992	Suspicious Action upon Submitted Information
8	0.0855	Links in <Meta>, <Script> and <Link> tags
9	0.0848	Using Free Domain Registration
10	0.0554	Request URL
11	0.0388	Google Index
12	0.0295	Using IP Address

From this table, we can see that a cyclical concept drift occurs because some features that are considered significant in the first data set, but drop from the second data set become significant again in the third data set. These features are, age of domain and Google index. In addition, some features become insignificant and dropped from the list achieved from the second data set. Those are, .htaccess redirect, PageRank and statistical-reports based feature. We will consider the third data set as a testing data set and pass it to the ESSNN in order to assess the effect of these changes on the ESSNN performance. The results show that the overall accuracy of the ESSNN drops to 92.02%. Although this accuracy rate is considered acceptable, we will create a new classifier and add it to the ESSNN in order to assess if it will improve the ESSNN performance. At the end of this experiment the ESSNN will include three classifiers. Table 6.7 shows the performance of the ESSNN as well as the considered DM algorithms on different threshold values.

The testing data set used for assessing the performance of the ESSNN as well as all considered DM algorithms is a combination of the testing data sets from the first, second, and third data sets. However, we notice that the same set of features has been selected when threshold value is set to 30% in all data sets (first, second, and third data sets) these are HTTPS and URL of Anchor. Therefore, no classifiers will be created at this threshold value because that will result in deriving a redundant classifier as confirmed from the results in the second data set.

Table 6.7 Confusion matrices (Third Data set)

Threshold	Algorithm	#Features	TPR	TNR	Precision	Recall	F1	Accuracy
Threshold = 1%	BN	12	92.00%	91.30%	92.10%	92.00%	92.05%	92.04%
	C4.5	12	92.20%	92.20%	92.40%	92.20%	92.30%	92.24%
	SVM	12	87.30%	85.20%	88.20%	87.30%	87.75%	87.35%
	LR	12	91.20%	90.20%	91.40%	91.20%	91.30%	91.22%
	RIPPER	12	90.40%	89.10%	90.70%	90.40%	90.55%	90.41%
	ESSNN	12	94.32%	96.79%	96.43%	94.32%	95.36%	95.81%
Threshold = 10%	BN	6	90.00%	91.10%	90.70%	90.00%	90.35%	90.00%
	C4.5	6	88.60%	87.20%	88.60%	88.60%	88.60%	88.57%
	SVM	6	87.80%	89.40%	89.30%	87.80%	88.54%	88.54%
	LR	6	89.70%	87.90%	89.80%	89.70%	89.75%	89.68%
	RIPPER	6	81.70%	84.50%	84.10%	81.70%	82.88%	81.66%
	ESSNN	6	90.00%	91.10%	90.70%	90.00%	90.35%	90.00%
Threshold = 20%	BN	4	92.90%	92.00%	93.10%	92.90%	93.00%	92.95%
	C4.5	4	92.30%	91.40%	92.50%	92.30%	92.40%	92.32%
	SVM	4	91.30%	90.90%	91.30%	91.30%	91.30%	91.28%
	LR	4	92.70%	91.90%	92.90%	92.70%	92.80%	92.74%
	RIPPER	4	91.10%	89.70%	91.50%	91.10%	91.30%	91.08%
	ESSNN	4	92.90%	91.30%	93.10%	92.90%	93.00%	92.95%

From Table 6.7 and Figure 6.4, we find that the ESSNN produces the best results when we add the classifier created at threshold=1% to the ESSNN. Not only that, but also the ESSNN achieves the best results in this experiment compared to the results achieved earlier, i.e. experiments on the first and the second data sets in terms of specificity (TNR), sensitivity (TPR), F1-Score, precision, and accuracy.

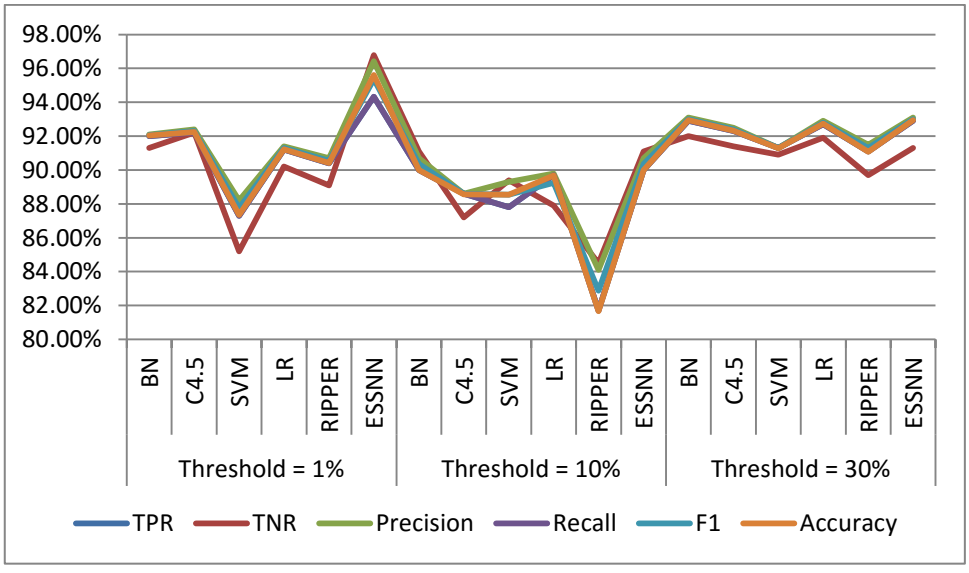


Figure 6.4 Performance of ESSNN and other DM algorithms (Third Data set)

In terms of F1-score, the ESSNN outperforms BN, C4.5, SVM, LR, and RIPPER by 3.31%, 3.06%, 7.61%, 4.06% and 4.81% respectively. In addition, the ESSNN

produces 3.77%, 3.57%, 8.46%, 4.59%, and 5.40% higher prediction accuracy than BN, C4.5, SVM, LR, and RIPPER respectively. That can be due to the fact that the ESSNN becomes more talented in dealing with different phishing websites that are designed using different features. In other words, the ESSNN reaches a stage of having several classifiers (three classifiers) where each of which is considered an expert in a particular part of feature space or concept. However, at the same threshold value we notice a slight improvement on the performance of the considered DM algorithms compared to the results achieved from the second data set. That is because of the emergence of a cyclical concept drift. RIPPER produces the worst results in this experiment among all other experiments when the threshold value is set to 10%. That is because RIPPER learns the rules greedily and prunes rules using incremental reduced error pruning method (Witten et al., 2011), which results in removing some rules that might be useful in determining the website class which in turn limits its ability in handling the virtual concept drift.

Overall, some features have shown to be more significant in predicting the phishing websites because they have been selected in all sub experiments when the threshold value is set to 1%. Those features are HTTPS, URL of anchor, adding prefix and suffix to domain, sub domain and multi sub domains, Using Free domain registration, request URL, links in <Meta>, <Script> and <Link> tags and web traffic.

In general, this set of experiments shows that the traditional DM algorithms that are commonly used for classifying phishing are not the appropriate choice for the application of phishing websites. On the other hand, the ESSNN is able to handle the dynamic nature of phishing.

The results achieved from the ESSNN in experiments 1, 2 and 3 when the threshold value is set to 1% (the classifier that produced at this threshold value is the classifier that has been added to the ESSNN in all cases, i.e. in experiments 1, 2 and 3) have been compared against other considered DM algorithms using the paired *t*-test in order to observe the significance in performance. The results show that the ESSNN algorithm produced a significant improvement over other considered DM algorithms (3 wins and 0 losses) with 95% confidence level when the epoch size is set to 500. This result confirms that the ESSNN is able to handle the virtual concept drift that occurring in phishing websites.

6.5. ESSNN vs Single Classifier based Stream Mining

6.5.1. Experimental Strategy

This set of experiments aims to compare the performance of the ESSNN in terms of handling the virtual concept drift that characterizes the phishing websites with two single classifier stream mining algorithms which are the VFDT (Hulten et al., 2001) and the HAT algorithms (Bifet & Gavaldà, 2009). The three collected data sets are merged together. For both considered algorithms, we use the default parameter settings. The splitting criterion utilised is IG. The splitting confidence δ is 0.001. Splitting confidence represents the allowable error in a split decision. The tie threshold τ is 0.05. Tie threshold is the value below which a split will be forced to break ties between two almost identically useful split features. Several grace period values γ are used in this set of experiments. Grace period (window size) is the number of instances that must exist to calculate the hoeffding bound which is used to decide on the splitting node in the tree.

6.5.2. Results from ESSNN and other Single Classifier algorithms

The experimental results demonstrated in Table 6.8 show that the HAT algorithm outperforms VFDT in most cases. The HAT algorithm produces on average 1.48% higher F1-score than VFDT. In addition, the HAT algorithm achieves 1.18% higher accuracy than VFDT. That can be attributed because the HAT algorithm uses a dynamic forgetting procedure that keeps the most relevant examples (Bifet & Gavaldà, 2009). Each node in the tree produced by HAT can determine which of the previous examples are relevant for it.

Table 6.8 Confusion matrices (ESSNN vs VFDT & HAT)

Algorithm	Grace Period (γ)	TPR	TNR	Precision	Recall	F1	Accuracy
VFDT	200	91.70%	91.00%	90.80%	91.70%	91.30%	91.32%
	400	88.40%	92.10%	91.60%	88.40%	89.90%	90.23%
	600	89.90%	91.70%	91.30%	89.90%	90.60%	90.78%
	800	85.60%	93.40%	92.70%	85.60%	89.00%	89.56%
	1000	85.10%	93.50%	92.70%	85.10%	88.80%	89.34%
	Average	88.14%	92.34%	91.82%	88.14%	89.92%	90.25%
HAT	200	91.20%	91.90%	91.70%	91.20%	91.40%	91.54%
	400	92.50%	90.70%	90.70%	92.50%	91.50%	91.55%
	600	92.4%	90.10%	90.20%	92.40%	91.30%	91.25%
	800	91.30%	91.20%	91.00%	91.30%	91.20%	91.27%
	1000	93.20%	89.90%	90.00%	93.20%	91.60%	91.52%
	Average	92.12%	90.76%	90.72%	92.12%	91.40%	91.43%
ESSNN		96.80%	97.00%	97.80%	96.80%	97.30%	95.61%

However, the ESSNN outperforms both algorithms in all cases, i.e. in different window sizes. That can be because of the forgetting mechanism employed by single classifier stream mining algorithms. For instance, although the window size has had a noticeable impact on the performance of the considered single classifier stream mining algorithms, it is difficult to decide on the optimal window size that keeps the most relevant examples and excludes the examples that represent the outdated concepts. In addition, some examples that are considered outdated and drop from the window may return to become

relevant in the future, which leaves the classification model susceptible to a cyclical concept drift. Figure 6.5 and Figure 6.6 show the performance of the VFDT and the HAT algorithms on different window sizes. However, although the examples used to train any classifier in the ESSNN are also forgotten, the knowledge achieved from such examples is maintained and used to classify examples that emerge from the past concept. This is the main advantage of the ESSNN and ensemble based classification approach over single classifier based stream mining algorithms. In general, this set of experiments confirm that the ESSNN is more effective than the considered single classifier stream mining algorithms in handling the concept drift that occurs in the phishing websites.

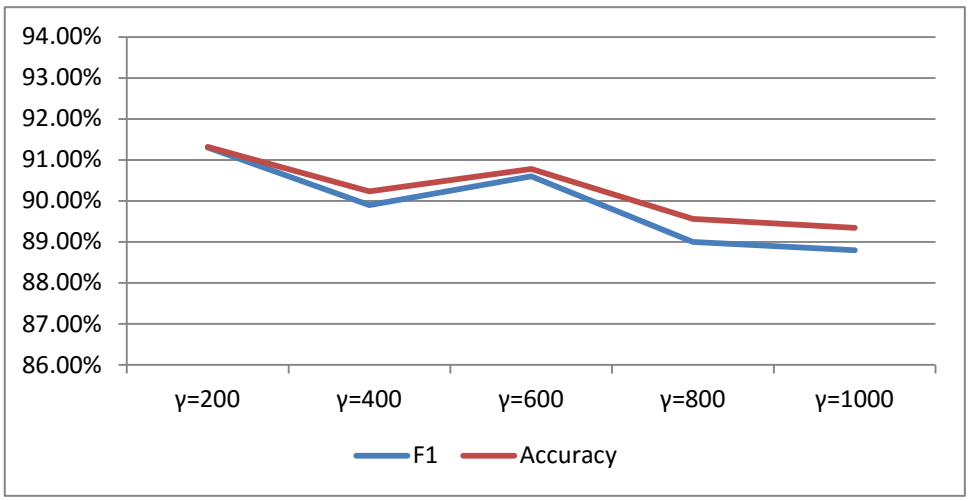


Figure 6.5 The impact of window size on VFDT performance

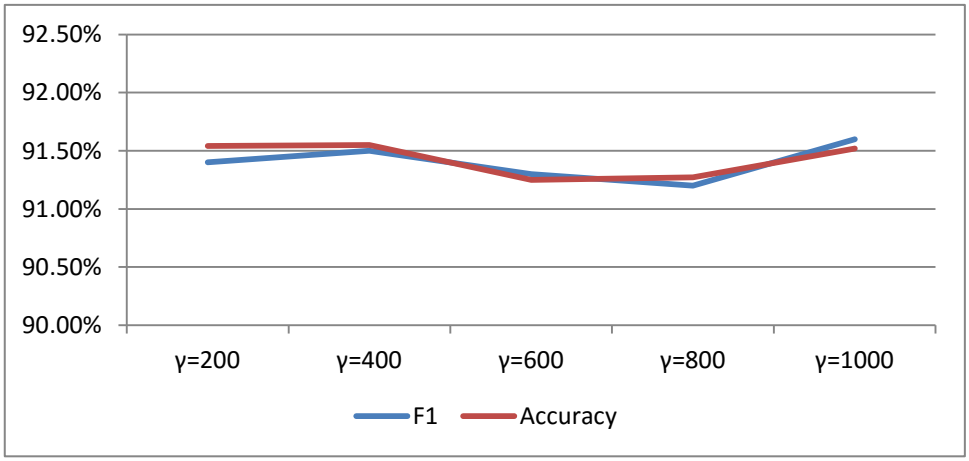


Figure 6.6 The impact of window size on HAT performance

6.6. Results Discussion

An intelligent method that reduces the human factor role in predicting phishing websites is an urgent need because phishing websites problem is expected to become more stylish and tactics used become more complicated. Such an intelligent method is expected to remove some of the burden from the internet users. DM methods - particularly classification techniques - become more viable in creating intelligent anti-phishing models (section 3.8). However, phishing websites pose a dynamic classification problem that are argued as being prone to virtual concept drift as shown in section 3.6. Therefore, any intelligent anti-phishing method should be updated regularly in order to keep pace with the latest phishing techniques and at the same time maintain what has been learned previously. Offline DM techniques are commonly used to tackle phishing (sections 3.8 and 3.9.7). Yet, these methods might not be the right choice to cope with the dynamic nature of phishing websites. For instance, the experimental results demonstrated in Tables 6.5 and 6.7 show that the performance of the considered offline DM algorithms drops when the virtual concept drift has appeared. This means that these methods may fail in predicting phishing if phishers change the combination of features when creating their phishing websites, which leaves the users susceptible to phishing attacks. Therefore, users may doubt the suitability of the Internet for commercial transactions. This throws a great obligation on the anti-phishing parties to produce intelligent tools that can produce high sensitivity (TPR) and specificity (TNR) rates. DM techniques that are capable of continuous learning shine on the horizon as a possible alternative.

Incremental learning has been debated as a viable solution for the dynamic domains as explained in section 2.9. Reviewing the literature, we have found few incremental learning methods that have been utilised in predicting

phishing (section 3.6). Window based approach is one of the incremental learning techniques as shown in section 2.10.1. The window size has a clear impact on the overall performance of the window based algorithms as revealed from the results shown in Table 6.8. However, the window based approach is criticized for being ineffective in handling the cyclical concept drift (Gama et al., 2014). For instance, in our experiments, we have found that the results obtained from the considered offline DM algorithms when the cyclical concept drift has occurred (Table 6.7) are very close to the results obtained from the considered window based algorithms (Table 6.8). Such results indicate that both techniques, i.e. offline DM technique and window based technique suffer from the catastrophic forgetting dilemma.

Ensemble based methods are another approach that can be used for incremental learning scenarios due in part to their ability to handle the cyclical concept drift. Such approach is able to learn new knowledge and at the same time maintain the previously learned knowledge. Therefore, the ensemble based approach can afford a balance between stability and plasticity. Looking at the experimental results demonstrated in Tables 6.3, 6.5, 6.7, and 6.8, we can see clearly that the ensemble based approach represented by the ESSNN is able to learn new knowledge and maintain the previously learned knowledge when applied to phishing websites classification problem. For instance, the ESSNN makes a positive jump in terms of the overall performance compared to other offline DM algorithms when the virtual concept drift has occurred (Tables 6.5 and 6.7). The experimental results provide convincing evidence that the ensemble based approach is more appropriate to contend with phishing websites elasticity. The good results achieved from the ESSNN can be attributed to several reasons. For instance, the classifiers that constitute the ESSNN are well structured in terms of number of hidden neurons and weights space. Although the average time needed to derive the classifiers using the

SSNN algorithm is higher than other considered algorithms (Table 5.5), the SSNN is able to produce good results in terms of average error-rate as shown in Table 5.3. One more reason for the good results achieved from the ESSNN is that it contains several classifiers each of which is considered an expert on different features space. Thus, it can deal with different phishing websites that are created using different features. The use of misclassified websites to learn a new classifier is another reason for the good results achieved from the ESSNN. Misclassified websites mean that the ESSNN might lack some knowledge. Yet, using the set of misclassified websites in deriving a new classifier gives the chance for the ESSNN to learn such knowledge.

However, the ESSNN regularly repeats the features selection process whenever a new data set becomes available in order to derive a new classifier and adds it to the ensemble. Nevertheless, the features that separate phishing websites from legitimate ones are not static. Although we have used many features in our study, we believe that phishers may come up with some novel features in the future. Yet, it is difficult to anticipate what these new features will be or when they might appear. As a result, it would be difficult to determine a specific size of the training data set that should be collected in order to create a new classifier and add it to the ensemble. Some techniques suggest revising the anti-phishing model periodically regardless of what the size of the training dataset is. For instance, in (Basnet et al., 2012), the authors suggest creating a new anti-phishing classification model on a weekly basis, whereas in (Ma et al., 2009), the authors recommend merging the newly collected data sets with the old one on a daily basis and the resulting data set is used to create a new classifier.

One of the improvements on the ESSNN might be utilizing a method that can extract the significant features on the fly. For instance, Deep Learning (Deng &

Yu, 2014) is one of the promising approaches that can extract features from raw-data. We believe that combining the ensemble approach with the deep learning technique can advance the ability of the ESSNN to cope with the virtual concept drift.

6.7. Chapter Summary

In this chapter, the ESSNN framework is applied to a vital web security problem where virtual concept drift occurs, that is the phishing websites classification problem. We started by collecting 30 features that distinguish phishing websites from legitimate ones. Two sets of experiments are conducted. The validation technique and the evaluation measures used in both sets of experiments are discussed. Three data sets are collected in different time spans. The results from the first set of experiments show that the considered DM algorithms (BN, C4.5, SVM, LR, and RIPPER) that are commonly used in designing anti-phishing models are hindered by their prediction accuracy if a virtual concept drift occurs. On the other hand, the ESSNN was able to effectively handle the virtual concept drift that occurs in phishing websites. The IG has been used to select the most effective set of features in each sub experiment conducted in the first set of experiments. Several threshold values are used to decide on the most effective set of features. Some features have shown to be more significant in predicting phishing websites, those are HTTPS, URL of anchor, adding prefix and suffix to domain, sub domain and multi sub domains, Using Free domain registration, request URL, links in <Meta>, <Script> and <Link> tags, and web traffic. Overall, the first set of experiments show that the ESSNN can lead to improved results compared to the results from any of the individual DM algorithms.

In the second set of experiments, we compare the performance of the ESSNN with two single classifier stream mining algorithms those are, the VFDT and

the HAT algorithm. Different grace period values (window sizes) are used. The results show that the window size has a noticeable impact on the performance of the single classifier stream mining algorithms. In general, the second set of experiments show that the ESSNN is more effective in handling the cyclical concept drift that occurs in phishing websites.

Overall, the experimental results show that the ESSNN furnishes a balance between stability and plasticity when applied to phishing websites.

CHAPTER 7

Conclusions and Future Work

7.1. Research Summary and Conclusions

This thesis investigated classification in data mining, and explored several issues related to classification in dynamic (non-stationary) domains, whereby the training data set is continuously evolving. Such data set may contain new knowledge which if not discovered may affect the overall performance of the classification model. These issues include virtual concept drift, catastrophic forgetting, and stability plasticity dilemmas. The result is a classification framework based on ensemble self-structuring neural network (ESSNN). ESSNN provided a balance between stability and plasticity since it has the ability to accommodate new knowledge, and does not forget the previously learnt one. The ESSNN creates a set of classifiers (ensemble) each of which is produced as a response to the virtual concept drift occurrence. Phishing websites pose a web security problem where virtual concept drift might occur. The proposed framework has been applied to phishing websites classification problem and showed competitive results when compared to several other data mining techniques. The classifiers that constitute the ensemble are created using an improved self-structuring neural network algorithm (SSNN). The SSNN algorithm simplifies constructing NN classifiers with minimum intervention from the user.

Many contributions evolved from this research and are summarized in the following subsections.

7.1.1. An Ensemble Self-Structuring Neural Network (ESSNN)

For dynamic domains where a virtual concept drift might occur such as phishing websites, the classification model should be adjusted continuously in order to maintain its performance. Therefore, a classification framework based on ensemble self-structuring neural network (ESSNN) that continuously obtains knowledge from evolving data sets is proposed in this thesis. The framework explores the advantage of regularly repeating the feature selection process to provoke new features into play. The framework supports reinforcement learning because it uses the set of misclassified instances at time t_i to derive a new classifier at time t_{i+1}. The classifiers that are added to the ensemble are created using the SSNN algorithm. The decision (assigning class value) of the ensemble is a collective decision, i.e. all classifiers in the ensemble contribute in producing the final decision. The framework does not only handle the virtual concept drift, but also furnishes a balance between stability and plasticity. The ESSNN (Chapter 4) has been accepted for publication and it will appear soon (Mohammad et al., 2016-A).

7.1.2. Phishing Websites Classification based on ESSNN

The ESSNN framework has been applied to phishing websites in order to assess the performance of the framework in handling the virtual concept drift occurring in phishing websites classification problem. Two sets of experiments are performed. The first set aims to compare the results when we treat phishing websites as a static classification problem with the results when we treat it as a

dynamic one. The results show that the ESSNN is able to effectively handle the virtual concept drift occurring in the phishing websites. Whereas the considered offline DM algorithms (BN, C4.5, SVM, LR, and RIPPER) that are commonly applied to phishing websites fail to address the virtual concept drift that occurs in phishing websites. This set of experiments revealed that some features are more significant in predicting phishing websites, those are HTTPS, URL of anchor, adding prefix and suffix to domain, sub domain and multi sub domains, Using Free domain registration, request URL, links in <Meta>, <Script> and <Link> tags and web traffic.

The second set of experiments aims to assess the performance of the ESSNN against two state-of-the-art single classifier based stream mining algorithms (window based algorithms) those are VFDT algorithm and HAT algorithm in handling the cyclical concept drift occurring in phishing websites. Although the considered window based algorithms produced good results, the ESSNN has shown to be more effective in handling the cyclical concept drift that characterizes phishing websites. In addition, this set of experiments shows that the window size has a noticeable impact on the performance of the considered window based algorithms. Overall, the experimental results showed that the ESSNN affords a balance between stability and plasticity when applied to phishing websites. The results of applying the ESSNN on phishing have been accepted for publication and will appear soon (Mohammad et al., 2016-A).

7.1.3. A Self-Structuring Neural Network Algorithm

As shown in section 2.4, two reasons may lead to the occurrence of the concept drift, i.e. virtual and real. In this thesis, we focus on the virtual one where the set of input attributes (features) changes over time. The virtual concept drift might occur naturally or intentionally. The intentional concept drift happens if

someone has the ability to comprehend how the classification model processes the input features in order to produce the final decision (assign class value). In this thesis, we make use of the black box nature of NN in order to tackle the virtual concept drift occurring deliberately. Nevertheless, NN models are normally created using a painstaking trial and error method, which often involves a tedious process. This problem has been tackled in this thesis by creating an improved NN structuring algorithm called Self-Structuring Neural Network (SSNN). Such an algorithm simplifies structuring NN classifiers. The SSNN is a family member of the constructive NN structuring approach. The algorithm is considered the cornerstone of the ESSNN because it creates the classifiers that are added to the ESSNN. As soon as a new data set is collected and after confirming the presence of a virtual concept drift, a new classifier is created using the SSNN algorithm.

In order to assess the performance of the classifiers that are derived from the SSNN algorithm, two sets of experiments have been conducted. In the first set, several experiments on a number of binary data sets from UCI repository have been done and the results have been compared against some well-known DM classification algorithms including C4.5, BN, LR, and FFNN. The data sets vary in size, i.e. number of instances, and number of attributes. In most cases, the experimental results show that the SSNN produced lower error-rate than other considered classification algorithms. In addition, the SSNN produced higher Relative Accuracy Rate (RAR) with compare to other considered classification algorithms, which reflects that the SSNN algorithm is able to produce well-structured NN classifiers. However, the average time needed for the SSNN to produce classifiers is higher than the average time needed for other contrasted algorithms. This may be due to the fact that the SSNN algorithm performs several sub-trainings in each training session before producing the final classifier.

In the second set of experiments, the SSNN has been applied to phishing websites data sets. The results reveal that the SSNN outperformed other considered classification algorithms in most cases. The SSNN algorithm produced the best results when using the Information Gain for feature selection. For instance, the average accuracy produced from the SSNN outperformed C4.5, BN, LR, and FFNN with margins of 0.79%, 1.38%, 1.39%, and 0.15% respectively when using the Information Gain. Overall, the experimental results show that the SSNN algorithm is able to produce good classifiers with good generalization ability.

The SSNN algorithm has been published in (Mohammad et al., 2013-B).

7.1.4. Providing a new Phishing Websites Classification Data Sets

By reviewing several previous studies related to classifying phishing using DM classification methods, and after analysing several phishing and legitimate websites, we managed to collect 30 different features that can distinguish phishing websites from legitimate ones. Several PHP and JavaScript codes have been written to extract these features. Finally, a new phishing websites data set has been donated to the University of California Irvine repository (UCI Repository) (Mohammad et al., 2015-B). To the best of our knowledge, this dataset is the first publically published structured data set. Such data sets have been used as a benchmark for evaluating phishing websites classification models as in (Abdelhamid, 2013) (Singh & Patil, 2014) (Abdelhamid et al., 2014) (Zeydan et al., 2014) (Abdelhamid, 2015) (Qabajeh & Thabtah, 2014) and (Mansour & Alshihri, 2015).

7.2. Future Work

In the near future, we intend to carry out several works; those are as follows:

7.2.1. A Multi-Class SSNN Algorithm

The SSNN algorithm proposed in this research is inclined towards binary classification domains. One of the future works is to improve the SSNN algorithm so that it can be applied to multi-class domains. However, this task needs in-depth investigations because classifying NN outputs into multiple categories is normally done by setting arbitrary value thresholds for discriminating one class from another. The one-vs-rest technique (Bishop, 2006) is one of the methods that can be used in creating NN classifiers for multi class domains. This technique involves training a single classifier per class with the examples of that class as positive examples and all other examples as negatives.

7.2.2. Identifying and Extracting Novel Features

In this research, we have collected 30 features that can separate phishing websites from genuine ones. We believe that some novel features might emerge in the future. Identifying and extracting novel features (input data attributes) is one of the most difficult and time-consuming phases when creating a classification model for dynamic domains where a virtual concept drift might occur as in phishing websites classification problem. Novel features might emerge for different reasons, for instance, the technical improvements in the web browser might be exploited by phishers as shown in twelveth feature in the Address Bar based Features, i.e. *The Existence of (HTTPS) Token in the Domain Part*.

Normally, this process is accomplished manually. The system designer keeps looking for any possible feature by analysing a set of legitimate and phishing

websites periodically. However, this technique is a long-winded process and the model designer has to dedicate a substantial amount of time to studying the changes in the input features.

Nowadays, Deep Learning (DL) (Deng & Yu, 2014) is a new and stimulating subfield of ML that attempts to sidestep the feature extraction and selection process and learn classifiers directly from the raw-data. In other words, DL has the ability to autonomously produce high-level representation from raw-data. Essentially, DL has been introduced with the aim of moving ML closer to one of its essential goals, i.e. Artificial Intelligence (Deng & Yu, 2014). The training process in DL is completely different when compared to traditional NN (Schmidhuber, 2015). For instance, the deep network consists of several hidden layers each of which is pre-trained with an unsupervised learning algorithm, which results in capturing more abstract attributes (features) from input data set (Schmidhuber, 2015). DL has been applied across a wide range of fields such as image classification (Krizhevsky et al., 2012), text classification (Zhang & LeCun, 2015), and signal processing (Yu & Deng, 2011).

A promising future work is to investigate how the ensemble method combined with Deep Learning can improve the classification accuracy and apply it to phishing websites.

7.2.3. Phishing in Smartphones and Mobile Devices

One of the future developments is to explore the need for anti-phishing solutions for applications in smartphones and mobile devices. Mobile devices or sometimes called handheld devices are available in various forms including Tablet PCs, Ultra-Mobile PCs and Smartphones. In 2016, the number of users who use handheld devices will surpass 2 billion users, doubling that of the number of personal computers (eMarketer, 2014). Trends in mobile application

development reveal an intense focus on some applications that are considered a fertile environment to start phishing attacks such as, social networking applications, mobile commerce applications, and mobile payment applications (UAB, 2014). In addition, there is a growing trend towards using mobile applications to conduct procurements. Mobile procurement applications facilitate completing procurement tasks anytime and anywhere especially for people who are most of the time out of their offices, away from their laptops, and constantly travelling. However, recent protection methods against phishing attacks in mobile applications are still far from ambition. Therefore, one of the future works is to develop a robust classification model to predict phishing in mobile technology and applications. Yet, the set of features that might be utilised to predict phishing websites on mobile phones might be different from those features used on laptops and personal computers. Therefore, one of the possible future works is to identify the most effective set of features that might be used in predicting phishing attacks on mobile phones.

Bibliography

Aaron, G. & Manning, R., 2014 A. APWG Phishing Reports. [Online] Available at: http://docs.apwg.org/reports/apwg_trends_report_q4_2014.pdf [Accessed 2 September 2015].

Aaron, G. & Manning, R., 2014 B. APWG Phishing Reports. [Online] Available at: http://docs.apwg.org/reports/APWG_Global_Phishing_Report_1H_2014.pdf [Accessed 2 September 2015].

Abdelhamid, N., 2013. Deriving Classifiers with Single and Multi-Label Rules using New Associative Classification Methods. PhD thesis.

Abdelhamid, N., 2015. Multi-label rules for phishing classification. Applied Computing and Informatics, 11(1), pp.29-46.

Abdelhamid, N., Ayesh, A. & Thabtah, , 2014. Phishing detection based Associative Classification data mining. Expert Systems with Applications, 41(13), p.5948–5959.

Abdelhamid, N. et al., 2012. MAC: A multiclass associative classification algorithm. Journal of Information & Knowledge Management, 11(2).

Abu-Nimeh, S., Nappa, D., Wang, X. & Nair, , 2007. A Comparison of Machine Learning Techniques for Phishing Detection. In The 2nd annual Anti-Phishing Working Groupse Crime researchers, eCrime '07. New York, NY, USA, 2007. ACM.

Aburrous, M., Hossain, M.A., Dahal, & Thabtah, , 2010 A. Experimental Case Studies for Investigating E-Banking Phishing Techniques and Attack Strategies. Cognitive Computation, 2(3), pp.242-53.

Aburrous, M., Hossain, M.A., Dahal, K. & Thabtah, F., 2010 B. Predicting Phishing Websites using Classification Mining Techniques. In The Seventh International Conference on Information Technology. Las Vegas, Nevada, USA, 2010 B. IEEE.

Aburrous, M., Hossain, M.A., Dahal, K. & Thabtah, F., 2010 C. Intelligent phishing detection system for e-banking using fuzzy data mining. Expert Systems with Applications: An International Journal, 37(12), pp.7913-21.

Afroz, & Greenstadt, R., 2011. PhishZoo: Detecting Phishing Websites by Looking at Them. In Fifth International Conference on Semantic Computing (September 18-September 21). Palo Alto, California USA, 2011. IEEE.

Akinyelu, A.A. & Adewumi, A.O., 2014. Classification of Phishing Email Using Random Forest Machine Learning Technique. Journal of Applied Mathematics, 2014(1).

Alallayah, K.M., AbdElwahed, W., Amin, M. & Alhamami, A.H., 2010. Attack of Against Simplified Data Encryption Standard Cipher System Using Neural Networks. Journal of Computer Science, 6(1), pp.29-35.

Alexa, 2011. Alexa the Web Information Company. [Online] Available at: http://www.alexa.com/ [Accessed 10 November 2011].

Alkhozae, M.G. & Batarfi, O.A., 2011. Phishing Websites Detection based on Phishing Characteristics in the Webpage Source Code. In International Journal of Information and Communication Technology Research., 2011.

Al-Momani, et al., 2011. An Online Model on Evolving Phishing E-mail Detection and Classification Method. Journal of Applied Sciences, 11(1), pp.3301-07.

Al-Ubaidy, , 2004. Black-box attack using neuro-identifier. Cryptologia (Taylor & Francis), 28(4), pp.358-72.

Amato, F. et al., 2013. Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine, 11(2), p.47–58.

APWG, 2003. [Online] Available at: http://www.antiphishing.org/ [Accessed 20 December 2011].

Arvandi, M., Wu, S. & Sadeghian, A., 2008. On the use of recurrent neural networks to design symmetric ciphers. Computational Intelligence Magazine, IEEE, 3(2), pp.42-53.

Authority of the House of Commons, 2013. E-crime, Fifth Report of Session 2013–14. London: he House of Commons The British House of Commons, Home Affairs Committ.

AWS, 2013. Alexa Top Sites. [Online] Available at: https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites- [Accessed 13 April 2013].

Basheer, I.A. & Hajmeer, M., 2000. Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods., 43(1), pp.3-31.

Basnet, R., Mukkamala, S. & Sung, A.H., 2008. Detection of Phishing Attacks: A Machine Learning Approach. Soft Computing Applications in Industry, 226(1), pp.373-83.

Basnet, R.B., Sung, A.H. & Liu, Q., 2011. Rule-Based Phishing Attack Detection. In The International Conference on Security and Management-SAM'11. Las Vegas, Nevada, 2011. IEEE.

Basnet, R.B., Sung, H. & Liu, Q., 2012. Feature Selection for Improved Phishing Detection. In 25th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems. Dalian, China, 2012. Springer Berlin Heidelberg.

Battiti, R., 1989. Accelerated Backpropagation Learning: Two Optimization Methods. Complex Systems, 3(4), pp.331-42.

BBC News, 2005. Jail for eBay phishing fraudster. [Online] Available at: http://news.bbc.co.uk/2/hi/uk_news/england/lancashire/4396914.stm [Accessed 20 October 2011].

Benitez, J.M., Castro, J.L. & Requena, I., 1997. Are artificial neural networks black boxes? IEEE Trans Neural Netw, 8(5), pp.1156-64.

Best SSL Certificates, 2011. SSL Certificate Reviews, Comparisons, and SSL Tutorials for Integration. [Online] Available at: http://www.bestsslcertificates.com/articles27.html [Accessed 8 March 2012].

Bifet, A. & Gavaldà, R., 2006. Kalman Filters and Adaptive Windows for Learning in Data Streams. In 9th International Conference, DS 2006. Barcelona, Spain, 2006. Springer Berlin Heidelberg.

Bifet, A. & Gavaldà, R., 2009. Adaptive Learning from Evolving Data Streams. Advances in Intelligent Data Analysis, 5772, pp.249-60.

Bifet, A., Holmes, G., Kirkby, R. & Pfahringer, B., 2010. MOA: Massive Online Analysis. The Journal of Machine Learning Research, 11(3), pp.1601-04.

Bishop, C., 2006. Pattern Recognition and Machine Learning. 1st ed. Springer-Verlag New York.

Bramer, M., 2013. Principles of data mining. 2nd ed. Springer-Verlag London.

Breiman, L., 1996. Bagging Predictors. Machine Learning, 24(2), pp.123-40.

Bright, M., 2011. MillerSmiles. [Online] Available at: http://www.millersmiles.co.uk/ [Accessed 10 December 2011].

Brown, K., 2006. A First Look at InfoCard. [Online] Available at: http://msdn.microsoft.com/en-us/magazine/cc163626.aspx [Accessed 20 January 2012].

Bubtiena, A.M., Elshafie, A.H. & Jafaar, O., 2011. Application of Artificial Neural networks in modeling water networks. In IEEE 7th International Colloquium on Signal Processing and its Applications (CSPA). Penang, 2011. IEEE.

Chandrasekaran, M., Narayanan, K. & Upadhyaya, S., 2006. Phishing email detection based on structural properties. In NYS Cyber Security Conference., 2006.

Chen, J. & Guo, C., 2006. Online Detection and Prevention of Phishing Attacks (Invited Paper). In First International Conference on Communications and Networking in China. ChinaCom '06. Beijing, 2006. IEEE.

Chen, M.-S., Han, J. & S. Yu, P., 1996. Data mining: an overview from a database perspective. IEEE Transactions on Knowledge and data Engineering, 8(6), pp.866-83.

Chou, N., Ledesma, R., Teraguchi, Y. & Mitchell, J.C., 2004. Client–side defense against web–based identity theft. In The 11th Annual Network and Distributed System Security Symposium (NDSS '04). San Diego, 2004. The Internet Society.

Cloudmark Inc, 2002. Cloudmark. [Online] Available at: http://www.cloudmark.com/en/home [Accessed 10 December 2011].

Cohen, W.W., 1995. Fast Effective Rule Induction. In In Proceedings of the Twelfth International Conference on Machine Learning. Tahoe City, California, 1995. Morgan Kaufmann.

Cohen, S. & Maimon, O., 2005. Reinforcement-Learning: An Overview from a Data Mining Perspective. In O. Maimon & L. Rokach, eds. Data Mining and Knowledge Discovery Handbook. Springer US. pp.469-86.

Coifman, R.R., 1992. Entropy-based algorithms for best basis selection. IEEE Transactions on Information Theory, 38(2), pp.713-18.

Cortes, C. & Vapnik, V., 1995. Support-Vector Networks. Machine Learning, 20(3), pp.273-97.

Cryptomathic, 2012. Two Factor Authentication for Banking, Building the Business Case. White Paper. Aarhus C, Denmark: Cryptomathic.

Dash, M. & Liu, H., 1997. Feature selection for classification. Intelligent Data Analysis, 1(1-4), pp.131-56.

Dean, B., 2013. Google's 200 Ranking Factors: The Complete List. [Online] Available at: http://backlinko.com/google-ranking-factors [Accessed 4 February 2014].

Deng, L. & Yu, D., 2014. Deep Learning: Methods and Applications. Foundations and Trends in Signal Processing, 7(3-4), pp.197-387.

Dhamija, R. & Tygar, J., 2005. The battle against phishing: Dynamic Security Skins. In The 1st Symposium On Usable Privacy and Security. New York, NY, USA, 2005. ACM Press.

Dhamija, R., Tygar, J.D. & Hearst, M., 2006. Why Phishing Works. In Proceedings of the SIGCHI conference on Human Factors in computing systems. New York, NY, USA, 2006. ACM.

Dharamkar, B. & Singh, R.R., 2014. Cyber-Attack Classification using Improved Ensemble Technique based on Support Vector Machine and Neural Network. International Journal of Computer Applications, 103(11), pp.1-7.

Dhillon, G., 2001. Information Security Management: Global Challenges in the New Millennium. Hershey, PA: IGI.

Dietterich, T.G., 2000. Ensemble Methods in Machine Learning. In The First International Workshop on Multiple Classifier Systems. London, 2000. Springer.

Domingos, P. & Hulten, G., 2000. Mining high-speed data streams. In The sixth ACM SIGKDD international conference on Knowledge discovery and data mining. New York, 2000. ACM.

Download.com, 2014. CNET Download.com. [Online] Available at: http://download.cnet.com/s/phishing/8/ [Accessed 5 May 2014].

Duffner, S. & Garcia, C., 2007. An Online Backpropagation Algorithm with Validation Error-Based Adaptive Learning Rate. In Artificial Neural Networks – ICANN 2007. Porto, Portugal, 2007. Springer Berlin Heidelberg.

eBay Toolbar's, 1995. Using eBay Toolbar's Account Guard. [Online] Available at: http://pages.ebay.com.au/help/account/toolbar-account-guard.html [Accessed 14 January 2012].

eMarketer, 2014. 2 Billion Consumers Worldwide to Get Smart(phones) by 2016. [Online Articel] eMarketer: eMarketer Available at: http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694 [Accessed 1 October 2015].

Engelbrecht, A.P. & Brits, R., 2001. A clustering approach to incremental learning for feedforward neural networks. In International Joint Conference on Neural Networks, 2001. Proceedings. IJCNN '01. Washington, DC, 2001. IEEE.

Executive-Order-13402, 2006. Executive Order 13402. [Online] Available at: http://www.gpo.gov/fdsys/pkg/FR-2006-05-15/pdf/06-4552.pdf [Accessed 14 March 2012].

Farid , D.M. et al., 2013. An adaptive ensemble classifier for mining concept drifting data streams. Expert Systems with Applications, 40(15), p.5895–5906.

Fayyad, U., Piatetsky-shapiro, G. & Smyth, P., 1996. The KDD Process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, 39(11), pp.27--34.

Fette, I., Sadeh, N. & Tomasic, A., 2007. Learning to detect phishing emails. In The 16th international conference on World Wide Web. Banff, Alberta, Canada, 2007. ACM.

Franklin, J., Perrig, A., Paxson, V. & Savage, S., 2007. An inquiry into the nature and causes of the wealth of internet miscreants. In The 14th ACM conference on Computer and communications security CCS '07. New York, 2007. ACM.

Freund, Y. & Schapire, R.E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), p.119–139.

Friedman, N., Geiger, D. & Goldszmidt, M., 1997. Bayesian Network Classifiers. Machine Learning - Special issue on learning with probabilistic representations, 29(2-3), pp.131-63.

FTC, 2015. Federal Trade Commission. [Online] Available at: http://www.ftc.gov/ [Accessed 3 September 2015].

Gaber, M.M., 2012. Advances in data stream mining. In Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. John Wiley & Sons, Inc. pp.79-85.

Gaber, M.M., Zaslavsky, A. & Krishnaswamy, S., 2005. Mining data streams: a review. ACM SIGMOD Record, 34(2), pp.18-26.

Gabralla, L.A., Mahersia, H. & Abraham, A., 2014. Ensemble Neurocomputing Based Oil Price Prediction. In The First International Afro-European Conference for Industrial Advancement AECIA. Addis Ababa, 2014. Springer International Publishing.

Gama, J. et al., 2014. A survey on concept drift adaptation. ACM Computing Surveys (CSUR), 46(4).

Ganatra, A., Kosta, Y.P., Panchal, G. & Gajjar, C., 2011. Initial classification through back propagation in a neural network following optimization through GA to evaluate the fitness of an algorithm. International Journal of Computer Science and Information Technology, 3(1), pp.98-116.

Garera, S., Provos, N., Chew, M. & Rubin, A.D., 2007. A framework for detection and measurement of phishing attacks. In The 2007 ACM workshop on Recurring malcode. Denver, Colorado, 2007. ACM.

Gartner IT Glossary, 2011. Gartner. [Online] Available at: http://www.gartner.com/it-glossary/data-mining [Accessed 30 May 2011].

Gastellier-Prevos, S., Granadillo, G.G. & Laurent, M., 2011. Decisive Heuristics to Differentiate Legitimate from Phishing Sites. In The 6th International Conference on Network and Information Systems Security (SAR-SSI). La Rochelle, France, 2011. IEEE.

General Assembly of Virginia, 2005. CHAPTER 827. [Online] Available at: http://leg1.state.va.us/cgi-bin/legp504.exe?051+ful+CHAP0827 [Accessed 2013 May 21].

Gill, P.E., Murray, W. & Wright, H., 1982. Practical optimization. John Wiley & Sons, Ltd.

Goodfellow, I.J. et al., 2015. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. Neural and Evolutionary Computing , 1(1).

Google Search Console, 2014. Site removed from the Google index. [Online] Available at: https://support.google.com/webmasters/answer/40052?hl=en [Accessed 15 February 2014].

Google-Safe-Browsing, 2010. Google Safe Browsing. [Online] Available at: http://code.google.com/p/google-safe-browsing/ [Accessed 11 December 2011].

Grbovic, M. & Vucetic, S., 2011. Tracking Concept Change with Incremental Boosting by Minimization of the Evolving Exponential Loss. Machine Learning and Knowledge Discovery in Databases, 6911, pp.516-32.

Greenwood, P.E. & Nikulin, M.S., 1996. A guide to chi-squared testing. New York: Wiley.

Gross, G., 2004. Senator introduces 'phishing' penalties bill. [Online] Available at: http://www.informationweek.com/phishers-would-face-5-years-under-new-bill/d/d-id/1030773?

Guyon, I. & Elisseeff, A., 2003. An introduction to variable and feature selection. Journal of Machine Learning Research, 3(7-8), pp.1157-82.

Hall, M. et al., 2011. Waikato Environment for Knowledge Analysis. [Online] Available at: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed 20 December 2011].

Hamker, F.H., 2001. Life-long learning cell structures--continuously learning without catastrophic interference. Neural Networks, 14(4-5), pp.551-73.

Han, J. & Moraga, C., 1995. The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira & F. Sandoval, eds. From Natural to Artificial Neural Computation. Malaga-Torremolinos, Spain.: Springer Berlin Heidelberg. pp.195-201.

Hansen, L. & Salamon, P., 1990. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1), pp.993-1001.

Harris Poll, 2006. Harris Poll: Taking Steps Against Identity Fraud. Harris Poll.

Herzberg, & Gbara, A., 2004. Protecting (even) Naive Web Users, or: preventing spoofing and establishing credentials of web sites. Technical Report. Ramat Gan: DIMACS Technical Report Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Laboratories America and Telcordia Technologies.

Hill, T. & Lewicki, P., 2007. STATISTICS: Methods and Applications. Tulsa, Oklahoma: StatSoft.

Hinton, G. et al., 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 29(6), pp.82-97.

Hoeffding, W., 1963. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301), pp.13-30.

Hoens, R.T., Polikar, R. & Chawla, N.V., 2012. Learning from streaming data with concept drift and imbalance: an overview. Progress in Artificial Intelligence, 1(1), pp.89-101.

Hornik, K., Stinchcombe, M. & White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), p.359–366.

Hossein, A.E., Amini, J. & Saradjian, M.R., 2003. Back propagation neural network for classification of IRS-1D satellite images. In Joint Workshop of High Resolution Mapping from Space. Tehran, 2003. Tehran University.

Hulten, G., Spencer, L. & Domingos, P., 2001. Mining time-changing data streams. In The seventh ACM SIGKDD international conference on Knowledge discovery and data mining. New York, 2001. ACM.

Insung, J. & Wang, G.-N., 2007. Pattern classification of back-propagation algorithm using exclusive connecting network. World Academy of Science, 36(1), pp.189-93.

Islam, M. et al., 2009. A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39(3), pp.705-22.

Jagatic, N., Johnson, A., Jakobsson, M. & Menczer, F., 2007. Social phishing. Communications of the ACM, 50(10), pp.94-100.

Jalal, F. & Peter, W., 1999. Digital Certificates: Applied Internet Security. Lewiston, NY, U.S.A.: Addison-Wesley.

James, L., 2005. Phishing Exposed. Syngress Inc.

Julie S., D., Mandy, H. & Cranor, L.F., 2007. Behavioral Response to Phishing Risk. In The Anti-Phishing Working Groups, 2nd annual eCrime researchers summite, Crime '07. New York, NY, USA, 2007. ACM.

Kalman, B.L. & Kwasny, S.C., 1992. Why tanh: choosing a sigmoidal function. In International Joint Conference on Neural Networks, 1992. IJCNN. Baltimore, MD, 1992. IEEE.

Kandel, E., Schwartz, J. & Jessel, T., 2000. Cell and Molecular Biology of the Neuron. In Siegelbaum. & Hudspeth, eds. Principles of Neural Science. McGraw-Hill Companies,Incorporated.

Kantardzic, M., 2011. Data Mining: Concepts, Models, Methods, and Algorithms. Second Edition ed. John Wiley & Sons.

Kaspersky Lab, 2013. Spam in January 2012: Love, Politics and Sport. [Online] Available at: http://www.kaspersky.com/about/news/spam/2012/Spam_in_January_2012_Love_Politics_and_Sport [Accessed 11 February 2013].

Kaufmann, M., Han, J. & Pei, J., 2011. Data Mining: Concepts and Techniques. Elsevier.

Keizer, , 2007. Phishers Beat Bank's Two-Factor Authentication. [Online] Available at: http://www.informationweek.com/phishers-beat-banks-two-factor-authentic/190400362 [Accessed 16 April 2013].

Kesari, V., Verma, L.K. & Tripathi, P., 2014. Image Classification using Backpropagation Algorithm. Journal of Computer Science, 1(2).

Khemphila, A. & Boonjing, V., 2011. Heart Disease Classification Using Neural Network and Feature Selection. In 21st International Conference on Systems Engineering (ICSEng). Las Vegas, NV, 2011. IEEE.

Kirda, E. & Kruegel, C., 2005. Protecting Users Against Phishing Attacks with AntiPhish. In The 29th Annual International Computer Software and Applications Conference. Washington, DC, USA, 2005. IEEE Computer Society.

Kriesel, D., 2007. A Brief Introduction to Neural Networks. dkriesel.com. Available at: http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf.

Krizhevsky, A., Sutskever, I. & Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C.J., Bottou, L. & Weinberger, K.Q., eds. Advances in Neural Information Processing Systems 25 (NIPS 2012)., 2012. Curran Associates, Inc.

Kubat, M. & Widmer, G., 1995. Adapting to drift in continuous domains (Extended abstract). In 8th European Conference on Machine Learning Heraclion. Crete, Greece, 1995. Springer Berlin Heidelberg.

Kumaraguru, P. et al., 2007. Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In eCrime '07 Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit. Pittsburgh, PA, USA, 2007. ACM.

Kwok, T.-Y. & Yeung, D.-Y., 1997. Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Transactions on Neural Networks, 8(3), pp.630-45.

Lam, L. & Suen, C.Y., 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. Systems, Man and Cybernetics, 27(5), pp.553-68.

Lee, W. & Xiang, D., 2001. Information-theoretic measures for anomaly detection. In IEEE Symposium on Security and Privacy, 2001. S&P 2001. Oakland, CA, 2001. IEEE.

LegitScript, 2007. The Leading Source of Internet Pharmacy Verification. [Online] Available at: http://www.legitscript.com/ [Accessed 18 June 2012].

Leyden, J., 2011. Free and subdomain hosting lets phishing sites live longer. [Online] Available at: http://www.theregister.co.uk/2011/04/27/phishing_trends_apwg/ [Accessed 13 Mar 2012].

Lichman, M., 2013. University of California, Irvine, School of Information and Computer Sciences. [Online] Available at: http://archive.ics.uci.edu/ml/index.html [Accessed 24 March 2011].

Linoff, S. & Berry, M.J.A., 2011. Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. John Wiley & Sons.

Littlestone, N., 1988. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. Machine Learning, 2(4), pp.285-318.

Littlestone, N. & Warmuth, M.K., 1994. The Weighted Majority Algorithm. Information and Computation, 108(2), pp.212-61.

Liu, , Deng, , Huang, & Y. Fu, A., 2006. An Antiphishing Strategy Based on Visual Similarity Assessment. Internet Computing, IEEE, 10(2), pp.58-65.

Ludl, , Mcallister, , Kirda, & Kruegel, , 2007. On the Effectiveness of Techniques to Detect Phishing Sites. Lecture Notes in Computer Science, 4579, pp.20-39.

Madhusmita, S., Dash, S.K., Dash, S. & Mohapatra, A., 2012. An approach for iris plant classification using neural network. International Journal on Soft Computing , 3(1).

Ma, L. & Khorasani, K., 2003. A new strategy for adaptively constructing multilayer feedforward neural networks. Neurocomputing, 51(1), p.361–385.

Malwarebytes, 2005. hoHosts. [Online] Available at: www.hosts-file.net [Accessed 22 March 2013].

Mannan, M. & Oorschot, P.C., 2007. Using a personal device to strengthen password. In 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007. Trinidad and Tobago, 2007. Springer Berlin Heidelberg.

Manning, C.D., Raghavan, P. & Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press.

Mansour, H.Y. & Alshihri, H.A., 2015. Adapting associative classification for detecting phishing websites. In The First Summit on Countering Cyber Crimes. Riyadh, 2015. Naif Arab University for Security Sciences.

MarkMonitor, 2013. MarkMonitor. [Online] Available at: https://www.markmonitor.com/ [Accessed 14 January 2013].

Markoff, J., 2010. Cyberattack on Google Said to Hit Password System. [Online] The New York Times Available at: http://www.nytimes.com/2010/04/20/technology/20google.html?_r=0 [Accessed 21 April 2015].

Marquardt, W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. Journal of the Society for Industrial and Applied Mathematics, 11(2), p.431–441.

Ma, J., Saul, L.K., Savage, S. & Voelker, G.M., 2009. Identifying suspicious URLs: an application of large-scale online learning. In The 26th Annual International Conference on Machine Learning. Montreal, Canada, 2009. ACM.

Masud, M.M., Woolam, C. & Latifur, J.G., 2012. Facing the reality of data stream classification: coping with scarcity of labeled data. Knowledge and Information Systems, 33(1), pp.213-44.

McAfee SiteAdvisor, 2006. SiteAdvisor. [Online] Available at: http://www.siteadvisor.com/ [Accessed 19 December 2011].

McCaffrey, , 2012. Classification and Prediction Using Neural Networks. [Online] Available at: https://msdn.microsoft.com/en-us/magazine/jj190808.aspx [Accessed 22 October 2012].

McCall, , 2007. Gartner, Inc. [Online] Available at: http://www.gartner.com/newsroom/id/565125 [Accessed 10 27 2011].

McCloskey, M. & Cohen, N.J., 1989. CatastrophicInterferenceinconnectionist networks. The psychology of learning and motivation, 1(24), pp.109-65.

McGrath, K.D. & Gupta, M., 2008. Behind phishing: An examination of phisher modi operandi. In The USENIX Workshop on Large-scale Exploits and Emergent Threats. Berkeley, CA, USA, 2008. USENIX Association.

Mesh, M., 2013. Tabnabbing Attack Method. [Online] Available at: http://www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_%28SET%29#Tabnabbing_Attack_Method [Accessed 21 Septemper 2013].

Microsoft-Support, 2012. Microsoft IE 9 anti-phishing. [Online] Available at: http://support.microsoft.com/kb/930168 [Accessed 19 December 2012].

Miller, R., 2005. More than 450 Phishing Attacks Used SSL in 2005. [Online] Available at: http://news.netcraft.com/archives/2005/12/28/more_than_450_phishing_attacks_used_ssl_in_2005.html [Accessed 8 March 2012].

Miyamoto, , Hazeyama, & Kadobayashi, , 2008. An Evaluation of Machine Learning-based Methods for Detection of Phishing Sites. In The 15th International conference on Advances in neuro-information processing ICONIP'08. Berlin, 2 October 2008. ACM.

Mizuno, S., Yamada, & Takahashi, , 2005. Authentication using multiple communication channels. In the 2005 workshop on Digital identity management. Fairfax, VA, USA, 2005. ACM.

Mohammad, R., McCluskey, T.L. & Thabtah, F., 2016-A. An ensemble self-structuring neural network approach and its application to to phishing websites. Neural Computing and Applications.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2012. An assessment of features related to phishing websites using an automated technique. In International Conference for Internet Technology And Secured Transactions, 2012. London, 2012. IEEE.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2013-A. Intelligent Rule based Phishing Websites Classification. IET Information Security, 8(3), pp.153-60.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2013-B. Predicting phishing websites based on self-structuring neural network. Neural Computing and Applications, 25(2), pp.443-58.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2013-C. Predicting Phishing Websites using Neural Network trained with Back-Propagation. In ICAI. Las Vigas, 2013-C. World Congress in Computer Science, Computer Engineering, and Applied Computing.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2015-A. Tutorial and critical analysis of phishing websites methods. Computer Science Review-ELSEVIER, 17(1), pp.1-24.

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2015-B. UCI-Phishing Dataset. [Online] Available at: https://archive.ics.uci.edu/ml/index.html [Accessed 26 March 2015].

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2015-C. Phishing Websites Dataset. [Online] Available at: http://eprints.hud.ac.uk/24330/ [Accessed 30 April 2015].

Mohammad, R.M., Thabtah, F. & McCluskey, L., 2016-B. An Improved Self-Structuring Neural Network. In Pacific Asia Knowledge Discovery and Data Mining Conference (PAKDD) 2016. Auckland, 2016-B. Springer (LNCS).

Møller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, 6(4), pp.525-33.

Mukkamala, S., Sung, A.H. & Abraham, A., 2003. Intrusion Detection Using Ensemble of Soft Computing Paradigms. Intelligent Systems Design and Applications, 23(1), pp.239-48.

Nahorney, B., 2015. The MessageLabs Intelligence Annual Security Report: 2009 Security Year in Review. [Online] Available at: http://www.symantec.com/content/en/us/enterprise/other_resources/intelligence-report-06-2015.en-us.pdf [Accessed 2 September 2015].

Netcraft Toolbar, 1995. Netcraft. [Online] Available at: http://toolbar.netcraft.com/ [Accessed 19 December 2011].

OpenDNS, 2006. OpenDNS. [Online] Available at: http://www.opendns.com/ [Accessed 12 February 2012].

Oza, C. & Russell, , 2001-A. Experimental comparisons of online and batch versions of bagging and boosting. In The seventh ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA, 2001-A. ACM.

PandaSecurity, 1990. Panda Security SL. [Online] Available at: http://www.pandasecurity.com/uk/ [Accessed 10 January 2012].

Pan, & Ding, , 2006. Anomaly Based Web Phishing Page Detection. In The 22nd Annual Computer Security Applications Conference (ACSAC). Miami Beach, Florida, USA, 2006. IEEE.

Parikh, D. & Polikar, R., 2007. An ensemble-based incremental learning approach to data fusion. IEEE Transactions on Systems, Man, and Cybernetics, 37(2), pp.437-50.

Paulin, F. & Santhakumaran, A., 2011. Classification of breast cancer by comparing back propagation training algorithms. International Journal on Computer Science and Engineering, 3(1), pp.327-23.

Peng, H., Long, F. & Ding, C., 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(8), pp.1226-38.

PhishTank Stats, 2012. PhishTank Stats. [Online] Available at: http://www.phishtank.com/stats.php [Accessed 8 February 2012].

PhishTank, 2011. PhishTank. [Online] Available at: http://www.phishtank.com/ [Accessed 10 December 2011].

Polikar, R., Upda, L., Upda, S.S. & Honavar, V., 2001. Learn++: an incremental learning algorithm for supervised neural networks. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 31(4), pp.497-508.

Qabajeh, I. & Thabtah, F., 2014. An Experimental Study for Assessing Email Classification Attributes Using Feature Selection Methods. In 3rd International

Conference on Advanced Computer Science Applications and Technologies (ACSAT). Amman, 2014. IEEE.

Qi, M. & Yang, C., 2006. Research and Design of Phishing Alarm System at Client Terminal. In Asian-Pasific conference on services computing (APSCC'06). Guangzhou, Guangdong, 2006. IEEE.

Quinlan, J., 1979. Discovering rules from large collections of examples: a case study. In Expert Systems in the Micro-electronic Age. Edinburgh, 1979.

Quinlan, J., 1986. Induction of decision trees. Machine Learning, 1(1), pp.81-106.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann.

Quinlan, J., 1998. Data mining tools See5 and C5.0. RuleQuest Research.

Quinlan, J.R. & Kaufmann, M., 1993. C4.5: Programs for Machine Learning. Machine Learning, 16(3), pp.235-40.

Ramanathan, V. & Wechsler, H., 2012. phishGILLNET—phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training. EURASIP Journal on Information Security, 2012(1).

Rasmussen, R. & Aaron, G., 2010. Global Phishing Survey: Trends and Domain Name Use 2H2009. [Online].

Riedmiller, M. & Braun, H., 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In The IEEE International Conference on Neural Networks. San Francisco, CA, 1993. IEEE.

Riley, M., Karl, J. & Chris, T., 2010. A Study of Early Stopping, Ensembling, and Patchworking for Cascade Correlation Neural Networks. IAENG International Journal of Applied Mathematics, 40(4), pp.307-16.

Rissanen, J., 2004. Minimum-Description-Length Principle. In Encyclopedia of Statistical Sciences. San Jose, California, 2004. John Wiley & Sons, Inc.

Rokach, L., 2010. Ensemble-based classifiers. Artificial Intelligence Review, 33(1), pp.1-39.

Ronald, D.J.C., Curtis, C. & Aaron, F.J., 2007. Phishing for user security awareness. Computers & Security, 26(1), pp.73-80.

Ronda, T., Saroiu, S. & Wolman, A., 2008. iTrustPage: A User-Assisted Anti-Phishing Tool. ACM SIGOPS Operating Systems Review, 42(4), pp.261-72.

Ross, et al., 2005. Stronger Password Authentication Using Browser Extensions. In the 14th Usenix Security Symposium. Baltimore,USA., 2005. USENIX Association.

RSA, 2012. RSA SecurID. [Online] Available at: http://www.rsa.com/node.aspx?id=1159 [Accessed 5 January 2012].

Rubner, Y., Tomasi, C. & Guibas, L.J., 1998. A Metric for Distributions with Applications to Image Databases. In The Sixth International Conference on Computer Vision ICCV. Bombay, 1998. IEEE Computer Society.

Rumelhart, E., Hinton, G.E. & Williams, R.J., 1988. Learning representations by back-propagating errors. In A.A. James & R. Edward, eds. Neurocomputing: Foundations of Research. Cambridge, MA, USA: MIT Press Cambridge. pp.696-99.

Salganicoff, M., 1997. Tolerating concept and sampling shift in lazy learning using prediction error context switching. Artificial Intelligence Review, 11(1-5), pp.133-55.

Sánchez-Maroño, N., Alonso-Betanzos, A. & Tombilla-Sanromán, M., 2007. Filter Methods for Feature Selection – A Comparative Study. In 8th International Conference in Intelligent Data Engineering and Automated Learning - IDEAL 2007. Birmingham, 2007. Springer Berlin Heidelberg.

Sanglerdsinlapachai, & Rungsawang, A., 2010. Using Domain Top-page Similarity Feature in Machine Learning-based Web. In Third International Conference on Knowledge Discovery and Data Mining., 2010. IEEE.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks, 61(1), pp.85-117.

Seipone, T. & Bullinaria, J.A., 2005. Evolving improved incremental learning schemes for neural network systems. In The 2005 IEEE Congress on Evolutionary Computation, 2005. Edinburgh, Scotland, UK., 2005. IEEE.

SEO, 2012. Search Engine Optimization (SEO). [Online] Available at: http://www.pagerank.net/ [Accessed April 2012].

Shamik, T., Singh, A.K. & Shukla, V.P., 2011. Statistical moments based noise classification using feed forward back propagation neural network. International journal of computer applications, 18(2), pp.36-40.

Shannon, C.E., 1948. A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), pp.379-423.

Sharifi, M. & Siadati, S.H., 2008. A phishing sites blacklist generator. In IEEE/ACS International Conference on Computer Systems and Applications, 2008. AICCSA 2008. Doha, 2008. IEEE.

Sheng, et al., 2010. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In CHI '10 Proceedings of the 28th international conference on Human factors in computing systems. New York, NY, USA, 2010. ACM.

Sheng, et al., 2007. Anti-Phishing Phil. [Online] Available at: http://cups.cs.cmu.edu/antiphishing_phil/ [Accessed 11 December 2011].

Sheng, S. et al., 2009. An Empirical Analysis of Phishing Blacklists. In Proceedings of the 6th Conference on Email and Anti-Spam (CEAS'09). Mountain View, CA, 2009. IEEE.

Shoemaker, L. & Hall, O., 2011. Anomaly detection using ensembles. In The 10th international conference on Multiple classifier MCS'11. Berlin, 2011. Springer-Verlag.

Singh, P. & Patil, D., 2014. Identification of Phishing Web Pages and Target Detection. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 3(2), pp.260-64.

Sivanandam, S.N., Deepa, S.N. & Sumathi, S., 2006. Introduction to Neural Networks Using Matlab 6.0. Tata McGraw-Hill Education.

Skrenta, R. & Truel, B., 2011. Open Directory Project. [Online] Available at: http://www.dmoz.org/ [Accessed 14 December 2011].

Sodiya, S., Onashoga, S. & Oladunjoye, B., 2007. Threat Modeling Using Fuzzy Logic Paradigm. Informing Science: International Journal of an Emerging Transdiscipline., 4(1), pp.53-61.

Spoofstick, 2005. spoofstick. [Online] Available at: http://www.spoofstick.com/.

Starr, J., 2012. Htaccess Made Easy: A Practical Guide for Administrators, Designers and Developers. Perishable Press. Available at: https://books.google.co.uk/books?id=NCyOMAEACAAJ.

Stevenson, A., 2014. Hackers phishing for source code from Microsoft, Apple and Oracle. [Online] V3.co.uk Available at: http://www.v3.co.uk/v3-uk/news/2374369/hackers-phishing-for-source-code-from-microsoft-apple-and-oracle [Accessed 25 April 2015].

StopBadware, 2010. StopBadware. [Online] Available at: https://www.stopbadware.org/top-50 [Accessed 12 February 2012].

Sugumaran, V., 2007. Intelligent Information Technologies: Concepts, Methodologies, Tools, and Applications. Oakland University.

Sullins, L., 2006. Phishing for a Solution : Domestic and International Approaches to Decreasing Online Identity Theft. Emory International Law Review, 20(1), pp.397-433.

Symantec Corporation, 2013. Internet Security Threat Report 2013. [Online] Available at: http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v18_2012_21291018.en-us.pdf [Accessed 8 May 2013].

Symantic, 1995. Verisign Authentication Services. [Online] Available at: http://www.verisign.com/ [Accessed 19 December 2011].

Thabtah, F., Cowling, P. & Peng, Y., 2005. MCAR: multi-class classification based on association rule. In The 3rd ACS/IEEE International Conference on Computer Systems and Applications., 2005. IEEE.

TrendMicro, 2013. Threat Reports. [Online] Available at: http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-cashing-in-on-digital-information.pdf [Accessed 2013 May 20].

TRUSTe, 1997. TRUSTe. [Online] Available at: http://www.truste.com/ [Accessed 10 May 2013].

Tsai, C.-J., Lee, C.-I. & Yang, W.-P., 2009. Mining decision rules on data streams in the presence of concept drifts. Expert Systems with Applications, 36(2), p.1164–1178.

UAB, 2014. The Future of Mobile Application. [Online] Available at: http://businessdegrees.uab.edu/resources/infographic/the-future-of-mobile-application/ [Accessed 1 October 2015].

Usama, F.M., Piatetsky-Shapiro, , Smyth, P. & Uthurusamy, R., 1996. Advances in knowledge discovery and data mining. 1st ed. Menlo Park, CA: MIT Press.

Wang, H., Fan, W., Yu, S. & Han, J., 2003. Mining concept-drifting data streams using ensemble classifiers. In The ninth ACM SIGKDD international conference on Knowledge discovery and data mining. Washington, DC, 2003. ACM.

Wang, J., Gao, K., Jiao, Y. & Li, G., 2009. Study on Ensemble Classification Methods towards Spam Filtering. In 5th International Conference, ADMA. Beijing, China, 2009. Springer Berlin Heidelberg.

Wang, H., Yu, P.S. & Han, J., 2004. Mining Concept-Drifting Data Streams. In O. Maimon & L. Rokach, eds. Data Mining and Knowledge Discovery Handbook. Springer US. pp.789-802.

Watson, D., Holz, T. & Mueller, S., 2005. Behind the Scenes of Phishing Attacks. [Online] The Honeynet Project Available at: http://www.honeynet.org/book/export/html/87 [Accessed 17 January 2012].

Wenyin, et al., 2005. Detection of Phishing Webpages based on Visual Similarity. In The 14th international conference on World Wide Web. New York, NY, USA, 2005. ACM.

White, H., 1990. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. Neural Networks, 3(5), pp.535-49.

Whittaker, C., Ryner, B. & Nazif, M., 2010. Large-Scale Automatic Classification of Phishing Pages. Regular. NDSS Symposium 2010.

WHOIS, 2005. WhoIS. [Online] Available at: http://who.is/ [Accessed 25 March 2012].

Widmer, & Kubat, , 1996. Learning in the presence of concept drift and hidden contexts. Machine Learning, Springer, 23(1), pp.69-101.

Widrow, B., Michael & Lehr, A., 1990. 30 years of adaptive neural networks. IEEE press, 78(6), pp.1415-42.

Wilamowski, B.M., 2009. Neural network architectures and learning algorithms. Industrial Electronics Magazine, IEEE , 3(4), pp.56-63.

Witten, I.H., Frank, E. & Mark, A.H., 2011. Data mining: practical machine learning tools and techniques with Java implementations. 3rd ed. Morgan Kaufmann.

WOT, 2006. Web of Trust. [Online] Available at: http://www.mywot.com/ [Accessed 24 January 2012].

Wu, M., Mille, R.C. & Garfinkel, S.L., 2006. Do security toolbars actually prevent phishing attacks? In Proceedings of the SIGCHI conference on Human Factors in computing systems. New York, NY, USA, 2006. ACM.

Wu, M., Miller, C. & Little, G., 2006. Web wallet: Preventing phishing attacks by revealing user intentions. In In Proceedings of the Symposium on Usable Privacy and Security (SOUPS). New York, NY, USA, 2006. ACM.

Xiang, G., Hong, J., Rose, C.P. & Cranor, L., 2011. CANTINA+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites. ACM Transactions on Information and System Security (TISSEC), 14(2), pp.1-28.

Yang, & Pedersen, J.O., 1997. A comparative study on feature selection in text categorization. In Fisher, D.H., ed. ICML-97, 14th International Conference on Machine Learning. San Francisco, 1997. ACM.

Yu, D. & Deng, L., 2011. Deep Learning and Its Applications to Signal and Information Processing. Signal Processing Magazine, IEEE, 28(1), pp.145-54.

Yu, L. & Liu, H., 2004. Efficient Feature Selection via Analysis of Relevance and Redundancy. The Journal of Machine Learning Research, 5(1), pp.1205-24.

Zette, K., 2010. Report: Google Hackers Stole Source Code of Global Password System. [Online] Available at: http://www.wired.com/2010/04/google-hackers/ [Accessed 18 April 2015].

Zeydan, H.Z., Selamat, A. & Salleh, M., 2014. Current state of anti-phishing approaches and revealing competencies. Journal of Theoretical and Applied Information Technology, pp.507-15.

Zhang, , Hong, & Cranor, , 2007. CANTINA: A Content-Based Approach to Detect Phishing Web Sites. In The 16th World Wide Web Conference. WWW '07. Banff, AB, Canada., 2007. ACM.

Zhang, X. & LeCun, Y., 2015. Text Understanding from Scratch. Technical Report. Cornell University.

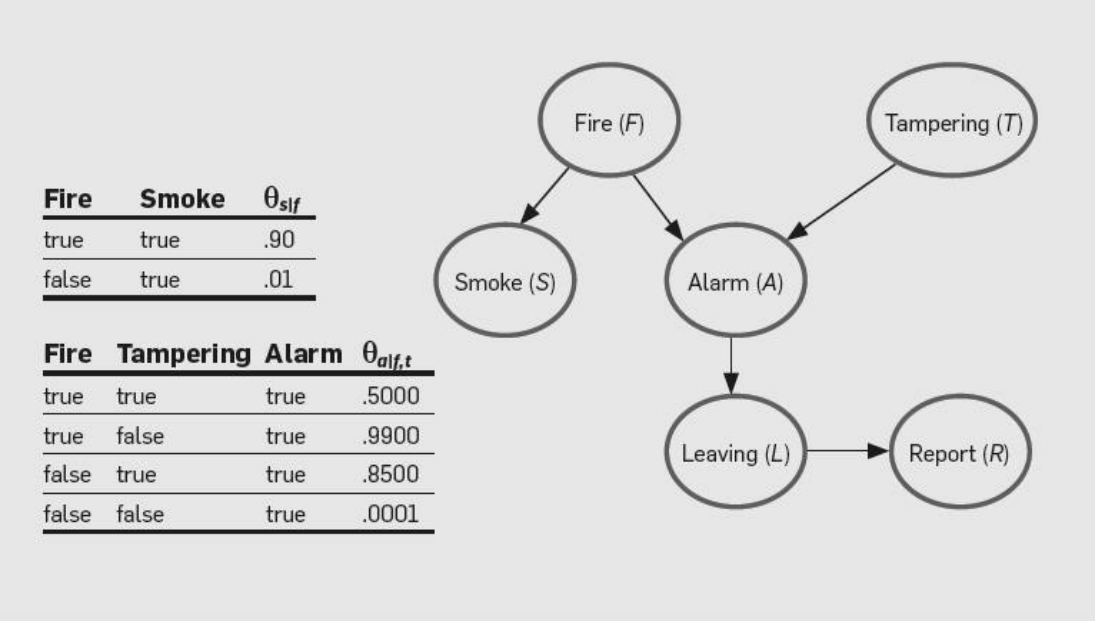
Zhang, Y., Lee, W. & Huang, Y.-A., 2003. Intrusion detection techniques for mobile wireless networks. Wireless Networks, 9(5), pp.545-56.

Appendix A

Common Classification Algorithms

▪ Bayesian Networks

Bayesian Network (BN) (Friedman et al., 1997) provides a comprehensible way to understand relationships between features (Sugumaran, 2007). BN affords an efficient method to represent relations between features and relatively allows rapid inference of probabilities. Figure 1 demonstrates a simple BN over six binary attributes. Normally, every BN has two elements: a directed acyclic graph termed a structure, and a set of Conditional Probability Tables (CPTs) (Sugumaran, 2007). The CPT tells how strong the relationship is between a variable and its parents in the network.



Fire	Smoke	$\theta_{s|f}$
true	true	.90
false	true	.01

Fire	Tampering	Alarm	$\theta_{a|f,t}$
true	true	true	.5000
true	false	true	.9900
false	true	true	.8500
false	false	true	.0001

Figure 1 A Bayesian Network with its Conditional Probability Tables

- **Decision Trees**

Decision Tree (DT) is a well-known classification method in which a set of rules are created and used to classify unseen data (Quinlan, 1979) (Quinlan, 1986) (Quinlan, 1993) (Quinlan, 1998). Two steps are recursively repeated to create a DT model:

1. Selecting the best feature for splitting the data set examples
2. Dividing the examples based on the value of the selected feature

The same procedure is applied iteratively until all the answers in all nodes give the same value; or the tree cannot be divided to any further extent (Quinlan, 1979). Picking the best attribute for splitting the data is done greedily to reduce the heterogeneity of the created partitions. The feature selection process should be done carefully because it might affect the distribution of the classes in each branch (Quinlan, 1979). Several feature selection algorithms might be utilised. The most commonly used technique (Bramer, 2013) among others is Information Gain (IG) (Shannon, 1948). Once the DT has been created, every single path from the root to the leaf nodes will produce a new rule. The connection between the root node and the leaf node is called rule antecedent. On the other hand, the rule consequent is the majority class that is associated with the leaf node. Several pruning mechanisms might be implemented to make the rules much simpler and to eliminate redundant rules (Quinlan, 1998).

- **Support Vector Machine**

Support Vector Machine (SVM) is a computationally expensive classification technique (Witten et al., 2011). SVM forms a hyperplane or set of hyperplanes in a high or infinite dimensional space that can be used for classification, regression, or other tasks (Cortes & Vapnik, 1995). A hyperplane separates a set of examples having different class memberships so that the examples are

divided by a clear gap that is as wide as possible as shown in Figure 2. SVM can handle multiple, continuous, and categorical variables.

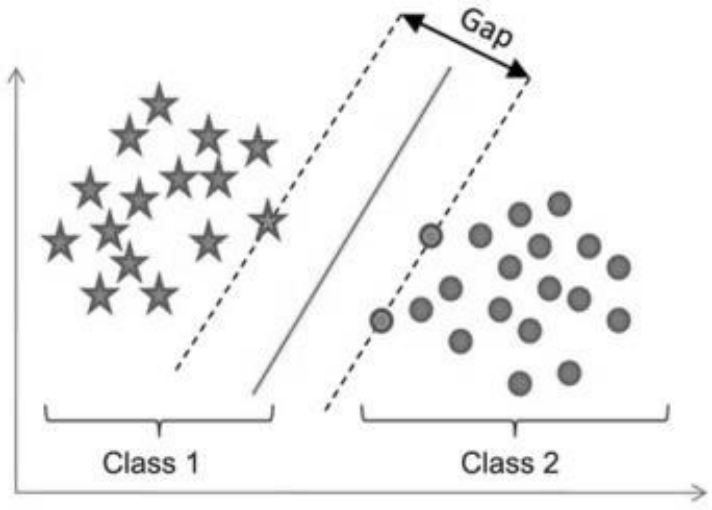


Figure 2 SVM for binary classification domain

- **Logistic Regression**

Logistic Regression (LR) is a statistical model widely used in domains where the class has binary values and it measures the relationship between the categorical dependent variable and one or more independent variables (Witten et al., 2011).LR gains its reputation due to its simplicity and interpretability. LR acts perfectly well when the relationship between the examples is almost linear (Hill & Lewicki, 2007).

- **Rule Induction**

Usually, a rule induction algorithm such as Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Cohen, 1995) divides the training data set with respect to class labels. Then, starting with the least frequent class set it builds a rule by adding attribute values to its body until the rule is perfect, i.e. the number of negative examples covered by the rule is zero. For each

candidate empty rule, RIPPER uses the Information Gain to pick the best attribute value in the data set and appends it to the rule body. RIPPER keeps adding attribute values until the rule becomes perfect. This phase is called rule growing. However, while building the rules, RIPPER uses an extensive pruning method to reduce rules redundancy and eliminate unnecessary attribute. The algorithm stops building the rules when any rule has a 50% error or in a new implementation of RIPPER when the minimum description length (MDL) (Rissanen, 2004) of the rules set after adding a candidate rule is larger than the one obtained before adding the candidate rule. The description length is defined by the number of the misclassified instances. After creating the rules set, one more pruning process is also done to produce the final classifier.

- **Artificial Neural Network**

An Artificial Neural Network (ANN) is a computerized model of the human brain and nervous system. Equally, the ANN and the human brain are consisting of a network of interrelated processing units called neurons (Kandel et al., 2000). From a biological perspective, the neuron controls the learning process and it consists of three elements: the dendrites, the soma or cell body, and the axon as per Figure 3. The dendrites receive inputs from the surrounding environment and is connected to tens of thousands of other neurons. The soma or cell body is the processing element that controls at which threshold the neuron will respond. During the learning process, the threshold value is adjusted several times to ensure that the soma becomes more talented. The axon is the elongated fibre that extends from the cell body to the terminal endings and transmits the neural signal. The larger the axon, the faster it transmits information (Widrow et al., 1990).

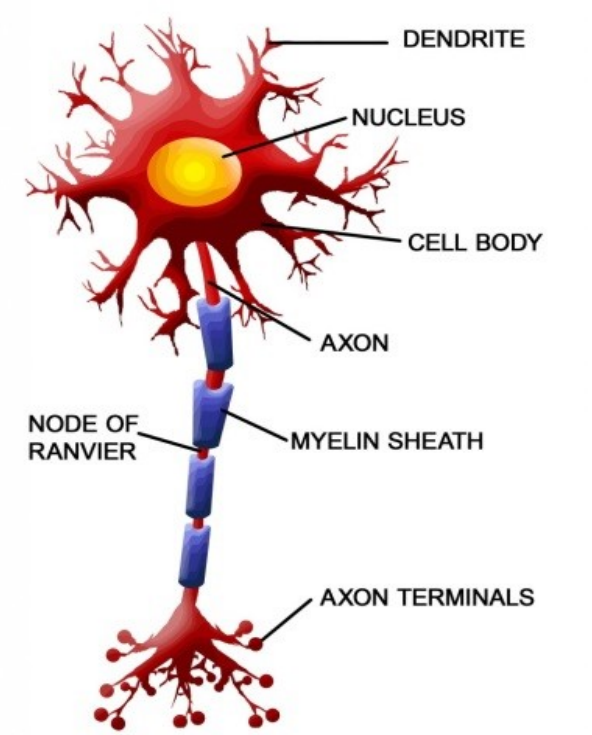


Figure 3 Biological Neuron

From the perspective of computer science, an ANN consists of a set of nodes as shown in Figure 4. Each node represents a neuron or a processing unit.

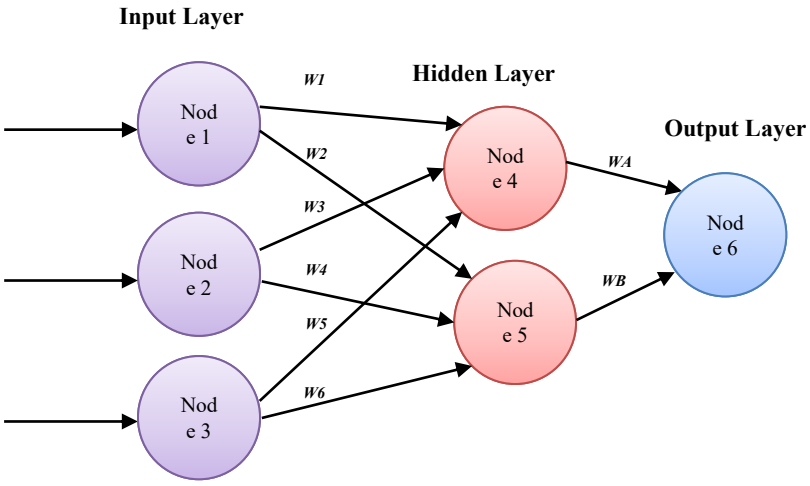


Figure 4 Simple feedforward ANN

The arrows represent the connection between the neurons. The node's output depends on a parameter called the connection weights *W* or synapse strengths

(Basheer & Hajmeer, 2000). Such weights are updated several times during the learning phase.

The most important elements in any ANN model are the connection weights and they denote the relative importance of each input to a specific processing unit. How the neurons are connected and the strength of these connections defines the behaviour of the ANN. Normally, the weights are adjusted using an error correction rule called delta rule or "*Widrow-Hoff learning rule*". Firstly, the error-rate is calculated as per equation 1.

$$err\ (i) = Actual\ value - Calculated\ value \qquad (1)$$

After calculating the error-rate, the result is passed to the delta rule shown in equation 2.

$$\Delta W(i) = \eta . err(i) . x(i) \qquad (2)$$

Where ΔW is the adjustments weight for the *i-th* neuron, x is the input value, and η is the learning rate. After calculating the adjustment weight, the old weights are updated as follows:

new weight = old weight + adjustment value

The learning rate controls the speed at which the ANN finds the best solution. If the learning rate value is very big then the learning will be fast but with the risk that the network will diverge from the best solution. On the other hand, a small learning rate might cause the network to take a very long time to converge to the best solution (Basheer & Hajmeer, 2000) (Widrow et al., 1990). Traditionally, the learning rate value is a static value and it does not change during the learning phase.

ANN models can be either supervised or unsupervised. The performance of any supervised ANN model can be easily measured by finding the difference between class values inferred by the ANN model and the true values in the

testing data set; such measurement is called cost function. One of the frequently used cost functions is the Mean Square Error (MSE). However, for unsupervised ANNs, there is no clear measurement indicator to evaluate the model. Two main architectures can be cited in supervised ANN (Wilamowski, 2009) those are, multi-layer feedforward neural network and recurrent neural network. Multi-layer feedforward NN processes the input variables in a unidirectional manner starting from the input layer towards the output layer. On the other hand, recurrent NN is a bidirectional data flow, since the values achieved from the output or hidden layer may propagate back to earlier layers through a feedback connections. These feedback connections could be either between neurons of different layers or a loop type self-connection. Several learning algorithms can be used in the feedforward ANNs such as Backpropagation (Rumelhart et al., 1988), Levenberg-Marquardt (Marquardt, 1963), BFGS Quasi-Newton (Gill et al., 1982), Resilient backpropagation algorithm (Riedmiller & Braun, 1993) and Scaled Conjugate Gradient backpropagation (Møller, 1993). However, backpropagation algorithm is the most commonly used and the simplest learning algorithm among others (Basheer & Hajmeer, 2000) (McCaffrey, 2012).

During their travel from the input layer to the output layer, the data set items are processed several times to learn the NN and update the connection weights between the neurons. For each layer, an activation function processes the data set items and then they are passed to the next layer. Several activation functions might be used, and they might differ from one layer to another. Firstly, the network calculates the net-input for each neuron by multiplying each input value with its corresponding weight, the result is passed to the activation function for further processing. The commonly used activation functions are as follows:

1. Hard limit activation function. Two types of this activation function have been introduced; binary activation function and bipolar activation function as per Figure 5 and Figure 6 respectively. The binary activation function produces 1 if the net-input is positive. Otherwise, it produces 0. On the other hand, bipolar activation function produces 1 if net-input is positive and -1 otherwise.

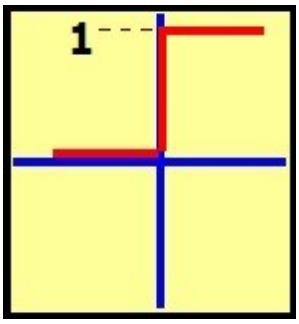


Figure 5 Binary hard limit

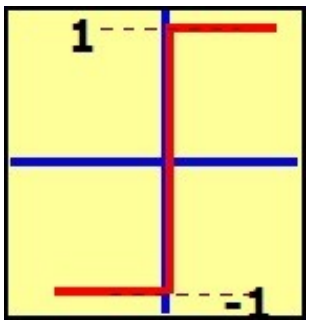


Figure 6 Bipolar hard limit

2. Piecewise linear activation function. This function produces 1 if the net-input is greater than or equal + $\frac{1}{2}$, and -1 if the net-input is less than or equal - $\frac{1}{2}$. However, if the net-inputs lies between positive and negative $\frac{1}{2}$ it will produce the net-input itself as shown in Figure 7.

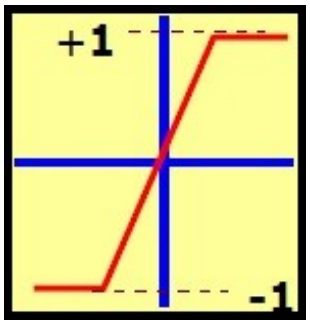


Figure 7 Piecewise linear activation function

3. Sigmoid activation function. This is the most commonly used activation function because of non-linearity and derivation simplicity (Han & Moraga, 1995). This function has an S shape as per Figure 8. This function is also

called logistic sigmoid or log-sigmoid. The function produces positive values only; and is calculated as per equation 3. Where ∑ is the net-input.

$$f(\Sigma)= \frac{1}{1+e^{-\Sigma}} \quad (3)$$

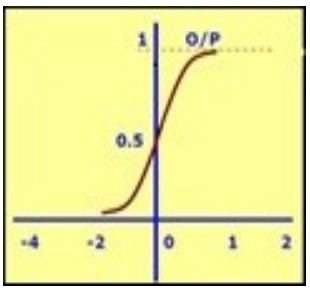


Figure 8 Log-Sigmoid

4. Hyperbolic tangent sigmoid activation function (TANH). This function ranges between -1 and 1 as per Figure 9. The TANH output is calculated per equation 4.

$$f(\Sigma)= \frac{e^{\Sigma}-e^{-\Sigma}}{e^{\Sigma}+e^{-\Sigma}} \quad (4)$$

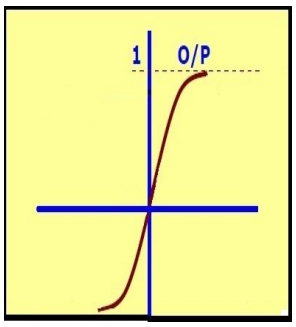


Figure 9 TANH function