TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ZAHRA ABBASZADEH

SUPERVISED FAULT DETECTION USING UNSTRUCTURED SERVER-LOG DATA TO SUPPORT ROOT CAUSE ANALYSIS

Master of Science thesis

Examiner: Prof. Moncef Gabbouj, Prof. Mikko Valkama
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 5th November 2014

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

ZAHRA ABBASZADEH: Supervised Fault Detection Using Unstructured Server-Log Data to Support Root Cause Analysis

Master of Science Thesis, 48 pages

November 2014

Major: Wireless Communication Circuits and Systems

Examiner: Prof. Moncef Gabbouj, Prof. Mikko Valkama

Keywords: fault detection, supervised learning, linear classification, SVM, log data

Fault detection is one of the most important aspects of telecommunication networks. Considering the growing scale and complexity of communication networks, maintenance and debugging have become extremely complicated and expensive. In complex systems, a higher rate of failure, due to the large number of components, has increased the importance of both fault detection and root cause analysis. Fault detection for communication networks is based on analyzing system logs from servers or different components in a network in order to determine if there is any unusual activity. However, detecting and diagnosing problems in such huge systems are challenging tasks for human, since the amount of information, which needs to be processed goes far beyond the level that can be handled manually. Therefore, there is an immense demand for automatic processing of datasets to extract the relevant data needed for detecting anomalies.

In a Big Data world, using machine learning techniques to analyze log data automatically becomes more and more popular. Machine learning based fault detection does not require any prior knowledge about the types of problems and does not rely on explicit programming (such as rule-based). Machine learning has the ability to improve its performance automatically through learning from experience.

In this thesis, we investigate supervised machine learning approaches to detect known faults from unstructured log data as a fast and efficient approach. As the aim is to identify abnormal cases against normal ones, anomaly detection is considered to be a binary classification. For extracting numerical features from event logs as a primary step in any

classification, we used windowing along with bag-of-words approaches considering their textual characteristics (high dimension and sparseness).

We focus on linear classification methods such as single layer perceptron and Support Vector Machines as promising candidate methods for supervised fault detection based on the textual characteristics of network-based server-log data. In order to generate an appropriate approach generalizing for detecting known faults, two important factors are investigated, namely the size of datasets and the time duration of faults. By investigating the experimental results concerning these two aforementioned factors, a two-layer classification is proposed to overcome the windowing and feature extraction challenges for long lasting faults. The thesis proposes a novel approach for collecting feature vectors for two layers of a two-layer classification. In the first layer we attempt to detect the starting line of each fault repetition as well as the fault duration. The obtained models from the first layer are used to create feature vectors for the second layer. In order to evaluate the learning algorithms and select the best detection model, cross validation and F-scores are used in this thesis because traditional metrics such as accuracy and error rates are not well suited for imbalanced datasets.

The experimental results show that the proposed SVM classifier provides the best performance independent of fault duration, while factors such as labelling rule and reduction of the feature space have no significant effect on the performance. In addition, the results show that the two-layer classification system can improve the performance of fault detection; however, a more suited approach for collecting feature vectors with smaller time span .needs to be further investigated.

PREFACE

This work has been conducted at the Department of Signal Processing of Tampere University of Technology in collaboration with TIETO within ICT SHOK D2I project..

I would like to express my deepest gratitude to my supervisor Prof. Moncef Gabbouj who trusted me in the first place giving me the opportunity to work in his research group while fully helping me throughout this process. This thesis could not be accomplished without his support.

I would like to thank Professor Mikko Valkama especially for revising my thesis and acting as an examiner.

My big appreciation goes to my co-supervisors Honglei Zhang and Dr. Stefan Uhlmann. Honglei supported and guided me throughout my research while Stefan provided me with constructive comments while writing the thesis. This thesis owes its existence to Stefan's great efforts, patience and feedback.

I would also like to thank my collaborators from Tieto, especially Harri Kukkasniemi, for providing me with the required dataset and helping me to conduct the research smoothly.

I do not have enough words to thank Sharareh Naghdi who has acted like my dearest sister and has made my life abroad memorable.

I would like to specially mention my friend Dr. Payman Aflaki for his help and suggestions on how to write this thesis.

Last but not least, my sincere gratitude goes my husband, Mehrdad, who experienced all ups and downs of my research. Moreover, I dedicate this thesis to my parents for their endless support and encouragement.

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

3G	Third generation of mobile telecommunications technology
4G	Fourth generation of mobile telecommunications technology
BCCH	Broadcast Control CHannel
FN	False Negative
FP	False Positive
FV1	Feature vector correspond to first window
FV1-L2	First feature vector of second layer
HDD	Hard Disk Drive
LTE	Long-Term Evolution
SVM	Support Vector Machine
LIBSVM	An implementation of Support Vector Machine
LIBLINEAR	An implementation of linear regression
TC	Text Classification
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
TTY	Unstructured log data including two days of network traffic data with one fault type
TTY-2	Unstructured log data including five days of network traffic data with five fault types
TP	True Positive
TN	True Negative
Wi-Fi	Wireless Fidelity

1. INTRODUCTION

Wireless communication is one of the big engineering success stories of recent years – not only from a scientific point of view, where the progress has been astonishing but also with respect to market size and impact on society. Cellular telephony is big evidence, as it is the biggest market segment, and has the highest impact on everyday lives. The popularity of mobile devices grows, as their offered capabilities and services increase. The growing scale of services they offer, diversity of devices connected to the network, introducing Heterogeneous Networks combining different Radio Access Technologies (3G, LTE (Long-Term Evolution), Wi-Fi (Wireless fidelity)) and more recently, variety of network topologies they can use, cause mobile communication networks to become more complex distributed systems [9]. High level of distributions results in producing huge amounts of data including, for example, measurements indicating radio interface efficiency and log data from different components [50].

As wireless systems are being increasingly used, it is becoming a challenge to debug and keep network operation stable, fault free, and secure. In complex systems, a higher rate of failure, due to the large number of components, has increased the importance of both fault detection and root cause analysis. Such failures can be caused by different sources such as power break down, hardware failure, software bugs, wrong configurations, human mistakes, and intrusive attacks. Root cause of a failure is the reason for which it occurs and root cause analysis is based on detecting and fixing the problem and preventing it from reoccurring.

The logs generated by network systems are generally time-series data streams contained messages that represent the status of a running system and other expansive information to support fault diagnosis. Therefore, syslog analysis plays an essential role for detecting failures in large systems [1]. However analyzing system logs manually is not realistic and practical since in a complex system various parts produce huge amounts of logs every day, which not all of them are interesting for error detection. Furthermore, software and hardware are developed and updated frequently leading to a considerable change of log files. As there is no standard structure for all log files, it is

infeasible to utilize a unique single log analyzer for a developing large system. Moreover, training many people to be as familiar with all system details is costly and time consuming [62]. To cope with these issues, machine learning techniques are introduced and exploited to automate the process of log analysis. Machine learning techniques can automatically process large scale data and improve their performance through learning patterns on a set of training data and then apply to new unknown data. In the field of intrusion detection, the advantages of machine learning techniques are their ability to detect unexpected problems and analyze all logs much faster than humans. Machine learning techniques do not rely on explicit programming. In addition, the developer of a machine learning system does not need to have any expert knowledge in log analysis; however, it could be helpful to find high quality features.

In this thesis, we investigate supervised machine learning approaches to detect known faults from unstructured log data simulated and provided by TIETO since these learning methods are fast and powerful in detecting known network faults and problems. Generally, supervised machine learning techniques make use of pre-existing knowledge and are first trained to generate models by using characteristics of labeled training data. Then the models are applied to identify faults and/or intrusions in unseen test data. But the preliminary and most important step in machine learning based fault detection systems is to extract numerical informative features from dataset since the performance of classification tasks is affected by representation of features.

To select appropriate feature extraction and classification methods, we focus on streaming and textual characteristics of log data as available dataset in communication networks. The used approach for feature extraction and collecting feature vectors in this thesis is sliding window (considering streaming characteristics of data) along with bag-of-words (considering textual characteristics of data). Regarding the textual characteristics of log data, linear classification techniques are proposed for learning algorithm. From various linear classification techniques, we choose to investigate two methods of single layer perceptron and Support Vector Machines (SVMs) with more focus on SVMs. Since SVMs are one of the best learning algorithms for binary classification that is relevant in this thesis to detect abnormal logs against normal ones [50, 83 and 1]. There are different implementations of SVMs. In this thesis, we compare the results of LIBLINEAR versus LIBSVM.

Our experiments are established in two phases based on usage of two different datasets. We explore the effects of using bag-of-strings instead of bag-of-words for feature extraction in order to reduce the dimension of feature space and different rules for labelling the feature vectors. The most challenging task in this thesis is to deal with long lasting faults. From classification point of view long lasting fault means that one class dominants the entire dataset while there is rare occurrences of another class. The problem with these kinds of dataset is low quality of their performance using standard detection approaches. To address this problem, this thesis proposes a two-layer classification with novel approaches for collecting feature vectors of each layer. The results show high performance of two-layer classification in compare to one layer classification for long fault durations. However, creating feature vectors for two layers are more time consuming. The rest of this thesis is organized as follows.

Chapter 2 provides background information and highlights the research relevant to this thesis. In this chapter, first the common approaches to extract features from text data and streaming data and previous work on feature extraction of log data are reviewed in general. Then the used classification methods in this thesis are introduced. In addition, two major detection approaches and various methods of intrusion detection used by researchers are given in this chapter.

Chapter 3 presents an overview of our approach for feature extraction, creating feature vectors and learning algorithm.

Chapter 4 introduces evaluation metrics and represents the experiments and results. In this chapter various factors that may affect the results are compared and the results of the chosen classification methods are illustrated.

Chapter 5 concludes and discusses limitations of supervised classification over unstructured network log-data.

2. BACKGROUND

Mobile networks have become interesting and popular networks in recent years due to the rapid increasing of wireless devices such as mobile laptop computers, PDAs and wireless telephones and growing scale of services they offer. As mobile networks become more popular and widely used, their operation, maintenance, security and in general, protecting them against anomalous network behavior to improve services and support this popularity, has become one of the major concerns. The anomalies can be defined as a data pattern, which is different from normal data and requires attention [74]. Consequently, detecting anomalous behavior is an indispensable task in network operation. Anomaly detection for communication networks is based on analyzing system logs from servers or different components in a network and determines whether there is any unusual activity. However, detecting and diagnosing problems in such huge systems is a challenging task for both developers and operators since the amount of information needed to process them goes far beyond the level that can be handled manually [83]. Therefore, there is an immense demand for automatic processing of datasets to extract the relevant data needed for detecting anomalies.

Generally, datasets in communication networks are represented in log format. The log format varies by types of components that generate it. The only parts of each log to correlate various types of unstructured log data are timestamps. Therefore, logs establish unstructured time series files, and event logs are non-numerical logs in which the messages contain vocabulary of terms (or phrases). In this thesis, the datasets to analyze are event logs and by terms of system log or syslog I mean event logs. Thus, to analyze the log files and collect feature vectors to apply machine learning techniques and detect the anomalies, we consider approaches used for both textual data and streaming data.

In this chapter, we first review feature extraction methods for text data and online streaming data. Then we look at text classification focusing on linear classification, as a

main approach applied to event logs. The last part of this chapter reviews previous anomaly detection methods for log data.

2.1 Feature extraction

Feature extraction is a core and preliminary step of any classification using machine learning techniques. Considering aforementioned characteristics of event logs, we need to combine the feature extraction methods of streaming data with textual feature extraction methods. In following a brief overview of feature extraction methods for both textual data and streaming data is represented., and then some researches on extracting features and mining the patterns of logs which could be used in machine learning techniques are reviewed.

2.1.1 Feature extraction methods for text documents

In text analysis such as natural language document classification, the main idea for feature extraction is to extract words from the raw text data and convert them into numerical features called term-based method, which gives a machine learning model a simpler and more focused view of the text. The most common way to address this issue is bag-of-words representation, which extracts numerical features from text content [1, 65]. In this technique, text data is considered to be a collection of words, and a dictionary is built by collecting all terms that occur at least once in a collection of documents. bag-of-words is a vector whose components represent the number of occurrence of each word in a document called term frequency (TF) while disregarding the position information of the words in the document. Normalization is applied to scale the term frequencies to values between 0 and 1 in order to measure the importance of a term in a document. In this scheme each individual components of term frequency vector or term weights vector is regarded as a feature [65].

Besides words, using phrases rather than words referred as n-grams may also be used [58] since a collection of words (unigrams) ignore any word order dependence and cannot consider phrases and multi-word expressions. Thus, in some cases, a collection of bigrams (n=2) or n-grams instead of unigrams is preferred, where occurrence of pairs or more consecutive words are counted [65].

TF-IDF (Term Frequency-Inverse Document Frequency) is also a commonly used feature in natural language processing (NLP) [40]. By using this method, the weight of terms that occur frequently in a document is low and the weight of rarely occurrence increases in order to improve the accuracy of classification [62, 56].

Feature selection is also an important issue in different classification methods, which is defined as selecting a subset of features from original features in order to reduce the dimension of text features and remove non-informative words from text data to improve learning performance [79].

The most common and effective feature selection method in text data is stop-word removal [8, 18]. In [84] a wide variety of feature selection methods in text categorization are compared and their experimental results discussed.

2.1.2 Feature extraction for online streaming data

A data stream is a massive real time sequence of data, which is continuous, ordered (by timestamp or arrival time), and fast changing. An issue concerning online streaming data processing is that to store an entire data stream or to scan through it several times is impossible due to its great volume. Sliding window approaches are a simple, widely used and standard way for feature extraction when dealing with streaming data. Moreover, since data streams have a natural temporal ordering, new data are often more accurate and more relevant than older ones [61].

For streaming (time-series) data processing, two types of sliding windows have been presented in different researches [61, 27]. The main approach for both is to isolate the range of continuous data to a sliding window, either with a fixed size of window containing the most recent T items, called a count-base or a sequence-based sliding window, or windows contain items from last t time units, called time-base or timestamp-based sliding window. Their performance is based on making a window classifier that assign a label from predefined class labels or ground truth to each feature vector extracted from input window of width w. Using this method, each sequence of data is segmented in temporal windows of fixed size of T items or time slices of t seconds in length defined as $L_i = < l_i,\ldots,l_{i+t-1}>$ that starts at time i. Next temporal window is defined by window shift of r as $L_{i+r} = < l_{i+r},\ldots,l_{i+r+t-1} >$. Then, a feature vector is built up

by collecting features from each window and labeled them with a predefined categories based on defined decision rule. The reason for popularity of this method is its simplicity to apply to any classical learning algorithm.

2.1.3 Feature extraction for syslog data

Feature extraction for system logs is problem dependent and there is no generally accepted standard in this area. For those kinds of logs contained numerical data, features are already provided that allows focusing on learning algorithm. But for time-series, meaningful non-numerical logs called event logs (indicating the state of systems), the features must be extracted since machine learning algorithm cannot process them directly. Even though many different methods have been represented in various study researches [83], this issue is still under investigation.

Considering the time-series characteristic of log files, sliding window methods have been widely used in many machine learning based anomaly detection techniques. [83, 76, 74, 4, 9]. However, they used different methodology to extract features. For example in [83], authors concentrate on two kinds of features, the *state ratio vector* using time-based window to analyze the behavior of the system over a certain period of time, and *message count vector* to collect problems concerning individual activities. Additionally, they applied TF-IDF method to its *message count vector* in order to improve the accuracy of detecting errors of logs. Different researchers have used different methods based on their applications. For example [74] represent the frequencies of 2-grams as features for network logs. Authors in [9] compare two one-class modelling techniques; one-class Support Vector Machines and a Hellinger distance-based one-class modeling in which the technique for extracting features is bag-of-words. Main concept on windowing combined with one of the feature extraction methods researches represented tools to extract features from log files [83, 4]. They introduce either preprocessing technique [83] or a tool to extract relative features more exactly [4] to improve the detection results. In [4] the semi unstructured time series database, Splunk, is represented as a tool for automatic event boundary detection that breaks the text stream into separate events exploiting the timestamps. It is used to index, search, and analyze massive datasets. The study [83] used programming to extract structured information from unstructured data logs by parsing them and specifying their important properties. Au-

thors represent two kinds of properties, identifiers variables to identify the program object which can take a large number of distinct values, and state variables that are labels to show a set of possible states an object could have in program and can take a small number of distinct values.

2.2 Classification

To detect network operational problems, we need to analyze the features extracted from log files and attempt to find well suited classification techniques in order to achieve high classification or detection accuracy. As log files resemble text document characteristics, we first study commonly used text classification methods, considering specifications of text documents and their feature vectors.

Classification in general is a machine learning technique to predict labels for data instances, and text classification (TC) is learning task, which assigns pre-defined category labels automatically to text data. Traditionally, learning methods are divided into two types: supervised learning and unsupervised learning. Supervised learning methods exploit labeled data, pairs of input objects and their corresponding output, to learn a classifier, which can be used to predict the output labels of new unknown data. While unsupervised learning methods do not require labeled training samples to learn a classifier, hence they can be used to model the input data based on their statistical properties.

Data can be in variant of single-label (binary classification), multi-label classification or multi-class classification. Binary classification involves two classes composed of relevant (positive) or not relevant (negative) items with respect to specific application where exactly one class must be assigned to each document. Multi-class problems refer to classification tasks with more than two classes while in multi-label problems a sample may be relevant to more than one class. Most of the research in text processing has been focused on binary classifications since it can be extended to multi-class as well as multi-label classifications. The strategy to deal with these problems is to break the problem into a set of binary classification problems, one for each class. Then apply all the binary classifiers to new data and make decision based on all prediction results [1, 15].

As text data contain a large number of words, the numerical feature vectors gained from text data are high dimensional and sparse (most feature valued are zero) vectors. Therefore, text classification needs special techniques to address the problem of high dimensional sparse data [1]. A successfully used method proposed by researchers' [1, 15, 1] to solve these problems is using linear classification algorithms.

In the following section, the main concept of linearity is defined. Then two well-known linear classifications, perceptron and SVM that are used in this thesis, are theoretically discussed.

2.2.1 Linear classifiers

The aim of a linear classifier is to divide two classes by a linear separator based on a linear combination of features. A formula to express the idea of linear classification is

$$y = W \cdot X + b \qquad (2.1)$$

in 2D the discriminant is a line, in 3D is a plane, and in nD it is a hyper-plane) where $X = (x_1,\ldots,x_n)$ is the feature vector (e.g., normalized word frequency vector), $W= (w_1,\ldots,w_n)$, is a vector of linear coefficients with the same dimension of the feature space, which called weight vector, b is a constant value that does not depend on any input value called bias value, and y is output, indicating class label. For a linear classifier, the training data is used to learn W and for classifying new data only W is needed.

The core idea of the *single layer perceptron algorithm* is to define the class label of any real-valued numerical input feature X_i, using the sign of the predicted function y_i from the discriminant function $y_i = W \cdot X_i + b$. Let us consider binary classification where class labels are either y=1 or y=0. Figure 2.1 shows a simple example in a 2-dimensional feature space. It illustrates two different classes and the separating plane corresponding to $W \cdot X + b = 0$.

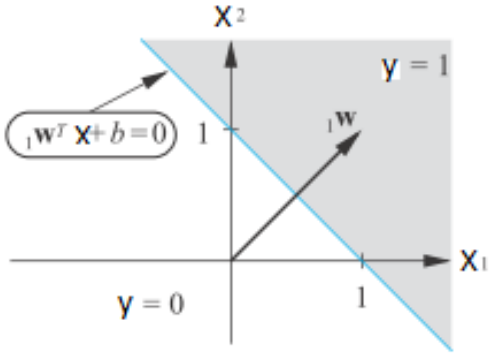


Figure 2. 1. *The sign of the projection onto the weight vector W yields the class label*

It is clear that the sign of the function W ·X + b determines the class label. Thus, the problem reduces to finding weights W with the use of training examples. The algorithm starts with initializing the weight vector randomly to equal values for all elements, and then updates these initial parameters when applying the current function on the training set makes mistake [68, 59, 11, 44, 69]. Learning rate α, where $0 < \alpha \leq 1$ can also be used to adjust the magnitude of the update, for example a too high learning rate makes the perceptron periodically oscillate around the solution.

The perceptron approximates a linear function, therefore if the training set is linearly separable; the perceptron is guaranteed to converge. In case the data is not linear separable then perceptron will not be able to find a good model to separate the data [44].

While the perceptron algorithm finds just any linear separation, Support Vector Machines (SVMs) [10, 81] are a kind of classifiers, which search for the best separator to have maximum margin between two groups of data according to some criterion. For example, consider two-class, separated training datasets of ‘x’ and ‘o’ that is illustrated in Figure 2.2 Comparing three different separating hyper-planes denoted by A, B, and C among many others, it is clear that the hyper-plane A provides better separation than the other two which are close to data points of one or both classes, and the normal distance of any of the data points from it, is the largest. Therefore, the hyper-plane A represents the maximum margin to closest points of ‘x’ and ‘o’.

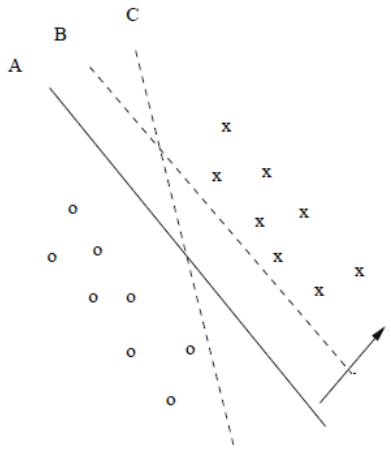


Figure 2. 2. *Finding the best separating hyper-plane*

The hyper-plane in SVM is constructed by using a subset of training data that are on the margin. These training data are referred to support vectors [84]. Figure 2.3 shows the margin and support vectors for a sample problem. [28] proposed that ”the ability of SVMs to learn can be independent of the dimensionality of the feature space”. This property causes SVMs to be able to apply for datasets with high dimensionality, if they are separable with a wide margin. In addition, SVMs can also be used for nonlinear classifiers using kernel functions [28].

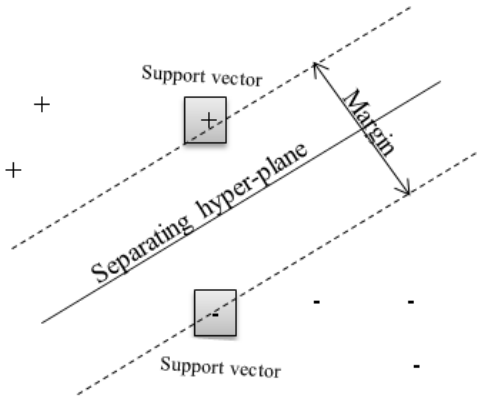


Figure 2. 3 *Margin and support vectors for SVMS*

For nonlinear separable samples, these techniques have the ability of mapping the original finite dimensional space into a higher dimensional space using kernels in order to have linearly separable samples. However, [19] mentioned that kernel functions are not so efficient for text classifications since the main assumption for kernels to be effective is that single words are not informative as high order word correlations. But, in some cases linear combination of word occurrences may provide this correlation. Therefore they can be effective for some special problems.

2.3 Literature Review

In general, there are various techniques for detecting network intrusions, signature (or misuse) detection, and anomaly detection [29, 30, 31, 32, 35 and 36]. From [36], both signature and anomaly-based detections are similar from conceptual operation point of view yet their main difference is in the nature of *attack* and *anomaly* terms. The term of *attack* refers to "a sequence of operations that puts the security of a system at risk" [36], while an *anomaly* is defined as "an event that is dubious from security point of view" [36].

In signature detection, the behaviour of a known intrusion or weak spots of a system are modelled to use for detecting known intrusions [29, 30, 31, 32, 33, 34, 35 and 36]. High accuracy of detecting known attacks with low false positive rate is the main advantage of this approach. But this approach is not able to detect unknown intrusions. In anomaly detection, normal behaviour of network is modelled and then it compares activities against the normal behaviour [29, 30, 31, 32, 34, 35 and 36]. The advantage of this approach is its ability to detect new intrusions. However, it cannot detect the intrusions that are not significantly different from normal activities, leading in high false positive rate [29, 30, 31, 32, 33, 34, 35, and 36].

Some research groups focused on signature (misuse) detection approaches using different techniques. For example, [40] introduced a prototype *Distributed Intrusion Detection System* (DIDS) that worked based on expert systems generating a set of rules that describe known attacks. Then the information from different components is analysed at a central location. [34] represented an example for state transition analysis approach. In this approach the process of intrusions are demonstrated as a series of state changes by using a graphical notation. Researches in [39] focused on developing a domain-specific

language called *behavioural monitoring specification language* (BMLS) to determine the relevant properties, from either normal behaviour of systems, or misuse behaviour associated with known attacks. They provide the STAT Tool Suite, which includes a language called STATL to describe the attack scenarios. But the problem of signature (misuse) detection is that it needs frequent rule base updates and signature updates. Nevertheless, this approach is not able to tackle the rapidly increased number of new attacks.

On the other hand, anomaly detection methods that model the normal network behaviour are relatively easy to perform, and effective in finding both known and unknown attacks. A vast number of researches have been performed on this topic using different methodologies [41, 42, 43, 44, 45, 46, 50, 51, 52, 53, 54 and 55]. These methods can be categorized into three different groups: statistical-based, specification-based and machine learning-based, which are briefly introduced next.

Statistical-based methods build operational profiles that describe normal behaviour of a system over a period of time. In general, normal profiles include probability distributions of different variables that represent the state of the system. Then a statistical distribution profile of new data is compared to the normal profile to distinguish significant differences and make decision based on this discriminant [41, 42 and 43]. The weaknesses of this method are that it ignores the temporal and multiple-variable correlation [67, 68].

Specification-based approaches are described in [44, 45 and 46]. In this approach, instead of modelling the normal activity, it builds a model based on specification of a secure operation. Accordingly, if an operation does not resemble this model then it is marked as an intrusion. This approach does not have the drawbacks of statistical-based methods; however it can be infeasible if the size of datasets is too big.

Machine learning can be defined as a programme or system that can learn from data and improve the performance over time. Thus, the strategy of a machine learning based method can change with new data. Necessity for labelled data to train a learning algorithm is its unique characteristic. But the ability of this technique to extract information directly from historical data without the need for manual work has attracted a lot of attention concerning intrusion detection. In addition, it can draw patterns over incom-

plete data and handle a large amount of data. Because of these reasons, variants of machine learning techniques have been applied to intrusion detection systems.

For example *Bayesian network* is a model that provides capability to capture relationships among variables of interest [36]. In general, this technique combined with statistical schemes is used for intrusion detections. Researchers in [50, 52, 53 and 54] provide different approaches combining Bayesian network model with variety of statistical values. But from [30], the problem with these methods is it depends on the assumptions about the behavioral model of the system. It means that the detection accuracy depends on the accuracy of chosen model. But finding an accurate model is a challenging task because of the complexity behavioral model within this system.

Clustering is another technique that works by grouping the data based on a given similarity or distance measure and characterizes anomalies considering dissimilarities [36]. A similarity measure is a key parameter in clustering to detect anomalies. For example, the k-nearest neighbor approach in [51] uses Euclidean distance to assign data points to a given cluster. Some sophisticated clustering also use fuzzy-k-mean and swarm-k-mean algorithms to improve the local convergence [55]. From [76] the advantage of clustering is its ability to learn from raw data in addition to detect intrusion in raw data without necessity of preprocessing and manual work. But for high dimensional data points it cannot provide the result with high accuracy.

Neural networks are human brain inspired approaches that have been employed for anomaly intrusion detection. Flexibility and adoptability to environmental changes are characteristics of these approaches; however there is not any learnable function [76] for making decision. Various approaches using neural networks for intrusion detection have been introduced by some research groups [59, 60 and 61]. In [62], the Anomalous Network-Traffic Detection with Self Organizing Maps (ANDSOM) was represented, which works based on monitoring a two dimensional Self Organizing Map (SOM) created for each network service. In training phase neurons are trained using normal network traffic. When feeding real time data to trained neurons, an anomaly is detected by comparing the distance of incoming traffic with a present threshold.

Support Vector Machines (SVMs) are another techniques involved in anomaly detections [69, 70]. Such techniques use one class learning techniques for SVM and learn a

region within the feature space with a maximum margin. Many researchers use variants of the basic technique (combined with other methods or different kinds of feature extraction methods) for detecting the anomalies in different fields such as computer and telecommunication networks. For example, [63] reports an improvement using SVM to the SOM approach used by [66]. In [65] authors have proposed a new robust approach of SVM for anomaly detection over noisy data. They have shown in their approach that testing time are faster since the number of support vectors is significantly less than compared to standard SVMs.

3. METHODOLOGY

The aim of this thesis is to investigate supervised machine learning approaches to detect known abnormal log behavior in log files as a fast and efficient approach. As mentioned in chapter 2, machine learning is the ability of a machine to improve its performance automatically through learning. A supervised learning technique needs labeled data in order to find a function or model that maps a sample into the class labels. Using labeled data during the training phase enables achieving clear feedbacks that help to learn quickly. Therefore, high efficiency and fast learning are advantages of supervised learning techniques. In addition, it is a powerful approach to detect known failures due to have robust patterns. Therefore, it can be a well suited approach and worth investigating for our special case detecting known faults.

In this approach, anomaly detection is considered to be a binary classification since the aim is to identify abnormal cases against normal ones. Thus, labelled data, as either normal or abnormal, is used for the learning phase in order to build detection models (profiles). Such models are employed for identifying the anomaly behaviors. The primary step in learning phase is to collect feature vectors as input data fed to training algorithm. In this approach, we collect feature vectors using sliding window, n-gram (bag-of-words), and word count to learn machine learning classification. From chapter 2, single layer perceptron and SVM are investigated as promising candidate methods of linear classifications for anomaly detection based on the textual characteristics of event logs used in this thesis. The most challenging phase of this investigation was big data including long lasting fault.

In this chapter, the methodology for generating the detection model is described. As illustrated in Figure 3.1 learning algorithm for building detection model contains feature extraction phase followed by model learning and testing phases to evaluate the accuracy of the detection. In Section 3.1, feature extraction approaches and mining the patterns of logs and prerequisites are defined. In Section 3.2, the approaches for detecting short duration faults as well as long lasting faults are described.

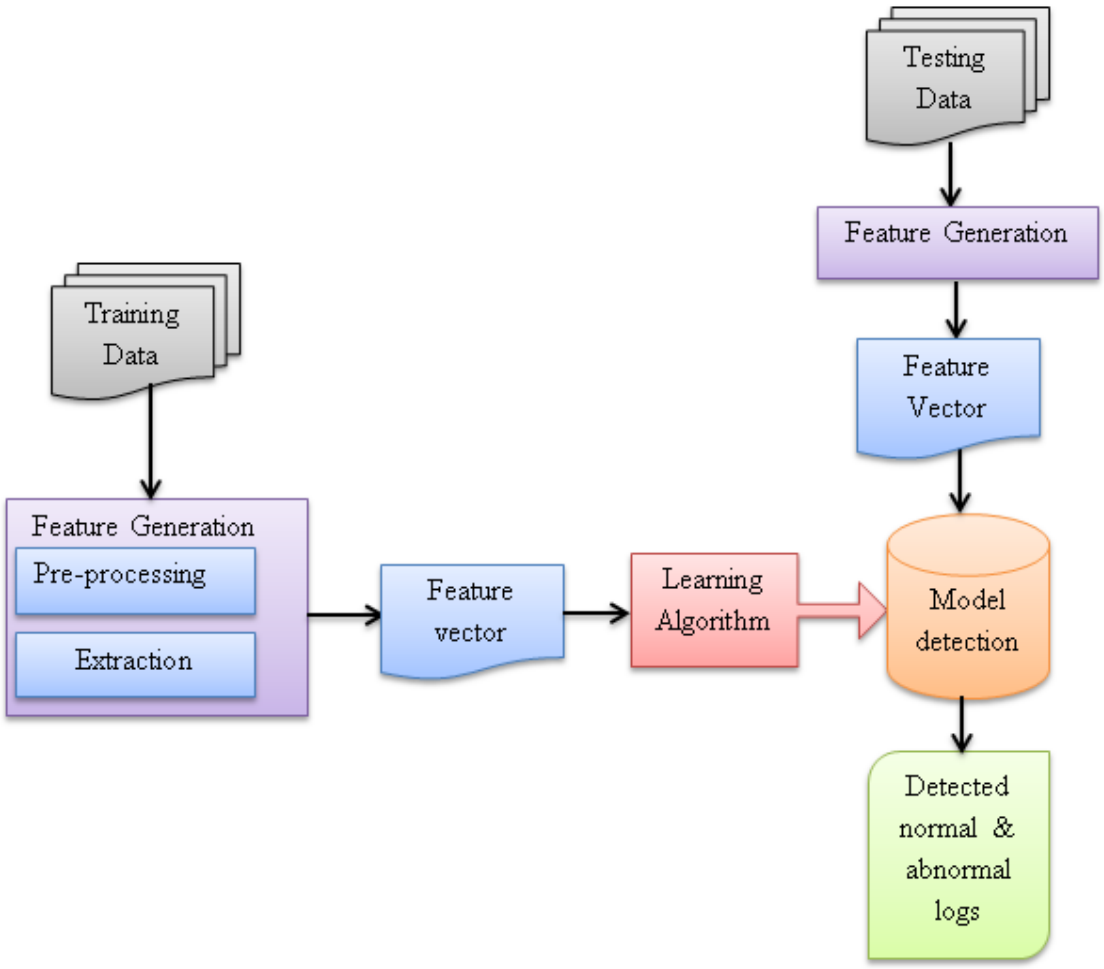


Figure 3. 1. *Overview of the learning algorithm*

3.1 Feature generation

Feature generation is an important part of any classification method. To achieve high quality anomaly detection, it is required to create high quality numerical features, indicating the log information which is understandable by machine learning classification. As reviewed in Chapter 2, several approaches for extracting features from log data have been well investigated in the literature. In this project we use windowing along with n-gram (bag-of-words) approach to extract features from event logs considering their textual characteristics. The following sub-sections describe the proposed approach investigated in this thesis.

3.1.1 Prerequisites

Ground truth: In order to analyse event log data and create labelled training samples, having ground truth is the first requirement. Mainly, ground truth dataset is human-expert knowledge based. It is defined as the labels associated with the data points to indicate if the data represents a problem or a normal case. Furthermore, ground truth can be used to evaluate the method by measuring the degree of match between ground truth labels (desired states) and actual ones obtained from classification methods. Figure 3.2 shows a part of ground truth dataset. Labels 1 indicate the fault logs, while labels 0 are associated with normal ones. In the cases that more than one fault is available in the event log data, one ground truth dataset is needed for each fault to allow processing each fault separately.

```
22.03.2014 18:29:57.525 0
22.03.2014 18:29:57.684 0
22.03.2014 18:29:57.880 0
22.03.2014 18:29:57.954 0
22.03.2014 18:30:00.023 0
22.03.2014 18:30:00.305 0
22.03.2014 18:30:01.148 1
22.03.2014 18:30:01.317 1
22.03.2014 18:30:01.578 1
22.03.2014 18:30:01.758 1
```

Figure 3. 2. *A part of ground truth dataset, fault logs labelled as 1 and normal logs labelled as 0*

Dictionary: As mentioned before and illustrated in Figure 3.3.a the format of logs (event logs) is not fixed. But there are some similar characteristics between all log messages. A log event typically has a timestamp with a fixed format representing the time at which the software has written the event. A log event also includes at least a text message containing English words, digits, and special characters. In a bag-of-words based feature model creating a dictionary, is a crucial prerequisites. By dictionary we means a collection of terms (i.e., words or phrases), as a reference for collecting numerical feature vectors. To do this, first message parts of all logs in a log data are tokenized using for example white-spaces and punctuation as token separators. In order to reduce

the dimension of dictionary, which relates to reducing the feature vector dimensionality, it is necessary to filter tokens and collect as informative words as possible. Filtering is done by removing digit numbers and special characters. Then by assigning an integer id for each unique English word the desired dictionary is created since English word messages are the most informative parts of logs. Figures 3.3.a and 3.3.b illustrate a log message and its corresponding tokens after filtering which are to create dictionary (collection of words)

```
14.03.2014 11:59:11.102 10.0.13.134 eNodeB-3475: NDD link up to DSA 6 node 3 #SpID=DBS_Internal_link
14.03.2014 11:59:11.747 10.0.13.181 eNodeB-3522: ALARM RAISE SP=38004 MC=/IPNManager-0
14.03.2014 11:59:12.902 10.0.16.216 eNodeB-4328: Load within capacity #SpID=DBS_Overload_over
```

a. A sequence of log messages

```
'ALARM'    'DSA'    'Load'    'NDD'    'RAISE'    'capacity'

'link'    'node'    'to'  'up'    'within'
```

b. Tokens after filtering and a sample of a dictionary as a collection of words

```
' ALARM RAISE'    ' Load within capacity'    ' NDD link up to DSA node'
```

c. Collected strings and a sample of a dictionary as a collection of strings

Figure 3. 3. *Sequence of logs and their corresponding tokens*

However, as the number of logs in a dataset increases, the dimension of dictionary and consequently dimension of feature vectors increases as well. As a result, the processing time significantly increases. To address this problem, using bag-of-strings inspired from n-gram methodology in text processing, instead of bag-of-words is proposed. To do this, each message line is converted to a message with reduced size by concatenating the final desired words (from aforementioned rules) of each line to create a string. As it is illustrated in Figure 3.3.c this approach causes the size of dictionary to reduce from the numbers of unique English words in a log data to the numbers of unique messages in that data, while each term in dictionary refers to a string instead of a word.

Labelling: As mentioned before, supervised learning needs labelled training data to build a detection model. Therefore it is necessary to assign label to each feature vector collected from raw data. To do this, time series dataset is segmented by windowing while segments are contained a set of labels (from ground truth dataset) associated to sequences of logs in that segment. Each segment is labelled based on the ratio of the numbers of normal logs to abnormal logs while threshold can be changed based on the series experimental results. For example in Figure 3.4 the rule for labelling is considered as one-fifth. It means if one-fifth of logs in each window are faults (labelled as 1) the feature vector corresponding to that window is labelled as fault.

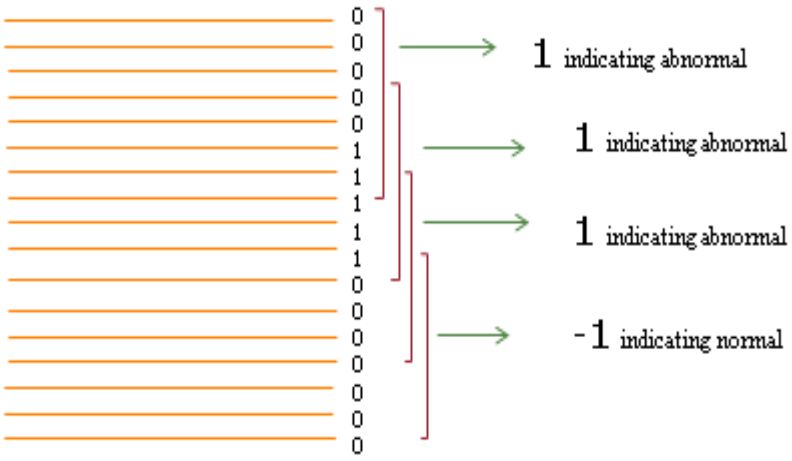


Figure 3. 4. *Assign label to each feature vector*

3.1.2 Feature Extraction

Our approach for collecting feature vectors is based on combining two strategies of sliding window and bag-of-words. Sliding window as a standard way to deal with streaming data is used to isolate sequential data. In this method, two properties of window size and sliding value need to be specified. The window size is used to limit a sequence of data used for processing to a certain range in time or number of logs. The sliding value is used to specify the execution condition of the processing. Whenever the process of collecting feature from certain window is performed, the sliding window is moved forward by a presumed value to specify next sequence. For instance, consider a

window of fixed size, 15 minutes, (in a time based windowing) and sliding value of 7 minutes (50% overlap) as it is shown in Figure 3.5 This window is placed at the beginning of the log file. All logs that fall in that window based on their time stamp are considered to be one sequence and create one feature vector. Then the window is moved forward 7 minutes and the next 15 minutes logs are made into a sequence. In the scenario that the duration of the faults is known (to have ground truth) the time based windowing is preferred.

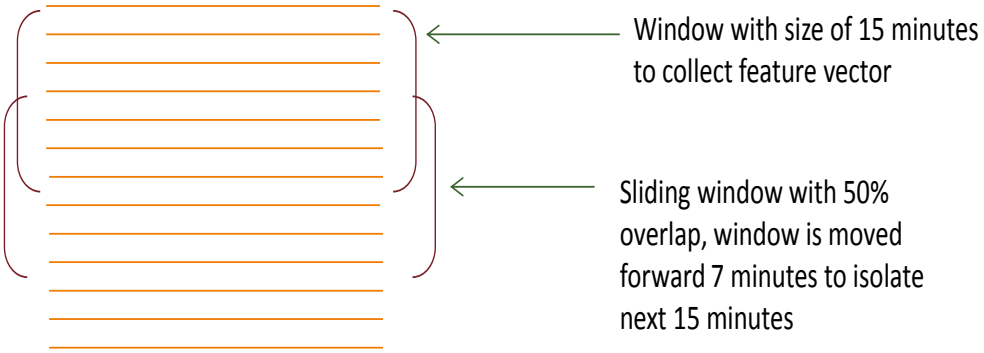


Figure 3. 5. *Sliding window for streaming data*

In bag-of-words based feature models, to collect feature vector associated with each segment, certain sequence specified by windowing, we need to tokenize the message parts of all logs in this window applying the same rules (tokenizing, filtering and collecting words or strings) used for creating dictionary. Then the number of occurrence of each term from dictionary in each segment is counted and collected as features to create feature vector (feature vectors and dictionary have equal dimension). This notation is called as term frequency in many documents.

For long documents, using raw counts directly particularly for linear classification is not efficient [77], because different numerical features in each feature vector may have different values. As mentioned in Chapter 2, in order to avoid those features with larger values being dominant, normalization is usually required. For normalization we consider the occurrence of each term versus the total number of occurrences of all terms in a feature vector to scale the term frequencies to values between 0 and 1.

3.2 Detection phase

Considering the textual characteristics of event logs, linear classification is used, as it is a well-suited method for text processing. Two important factors, the size of datasets and the time duration of faults, are investigated in order to generate an appropriate approach generalizing for detecting known faults. In following sub-sections, first, general approach; learning phase and testing phase used for short fault duration is described. Then in Section 3.2.1, we deal with feature extraction and classification methods for long faults.

3.2.1 General approach – short faults

Fault detection for datasets including short faults like any classification algorithm is implemented in two steps, *learning phase* and *testing phase*. The major task of learning phase is to build detection model using training dataset that provides information for detecting anomaly behaviors. In testing phase the performance of learning algorithms and feature extraction methods is evaluated. The testing phase algorithm employs the detection model to classify new dataset that are unknown to the algorithm. Then, detected labels are compared to the actual ones to estimate the performance of detection algorithm.

3.2.2 Long fault duration – two-layer classification

Prior to this sub-section, an overall technique for fault detection was described. But for long lasting faults where a large number of logs can be labeled as faults, the mentioned approach is not able to successfully detect such abnormality. In practice, a special case can be when a long fault is repeated several times throughout a dataset while many of them overlap with each other. Such scenario will result in fault reporting for majority of the logs. Two-layer classification with distinct approaches for extracting features in each layer is presented as a solution to address this kind of problem.

Two-layer classification comprises of two layers. The first layer includes two classifiers in parallel. For convenient understanding, we called them based on their characteristics, middle classification and start classification. Middle classification focuses on all abnormal logs while start classification concentrates on starting lines of each fault repe-

tition. The approach for middle classification is the same as previous ones. Feature vectors are collected using the approach mentioned in Section 3.1 while the best detection model from learning phase is required and saved to be used in the second layer.

As in start classification the focus is on starting lines of each fault occurrence, a different approach of extracting features is proposed for this type of classification. As it is shown in Figure 3.6 for extracting features and collecting feature vectors, the windowing is started from the first line of dataset. The window size and sliding value must be the same as what was used for middle classification. However, the windows including first line of each fault must be distinct without any overlap. It is needed to assign a specific window with the same size of others to the range of logs, including starting line of fault, in such a way that the large part of the window includes the logs which come after starting line of fault. For example by considering window size of 15 minutes, we specify the window to start 2 minutes before timestamp of starting line and to terminate 13 minutes after starting line of each repetition of the fault. Then feature vector associated to each window is created using previous method (bag-of-words or bag-of-strings). For labelling, windows containing starting line of each fault are labeled as abnormal and all other windows are labeled as normal. Figure 3.6 shows that discarding some lines is inevitable in this approach in order to avoid an overlap between windows including start line of each fault and other windows. After collecting feature vectors, detection models is built from learning phase and the best one is saved from testing phase. As an important point, detection model for both middle and start classifications must be created using the same classification method.

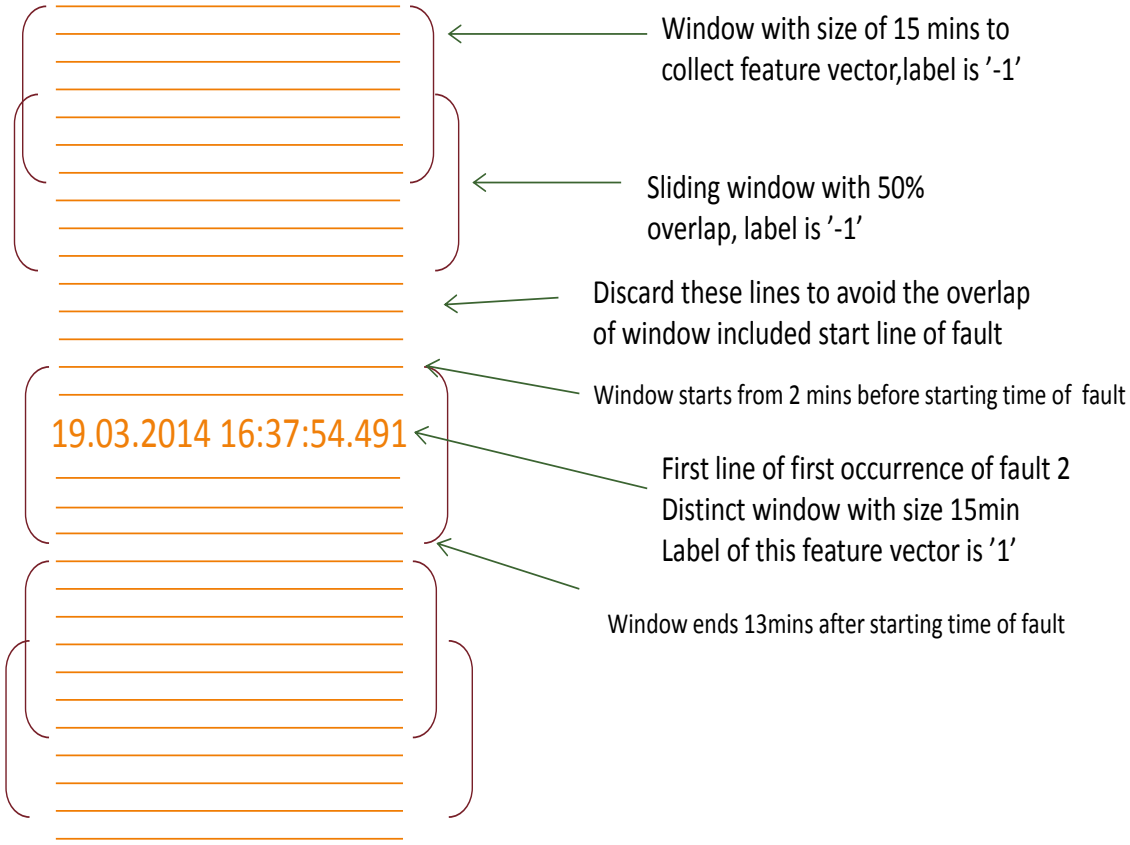


Figure 3. 6. *The approach of feature extraction for start classification of first layer*

The approach of extracting features for second layer is completely different from what has been used so far. To collect feature vectors of second layer, two detection models produced from first layer are used exploiting the collecting feature vectors approach described in 3.1 Section. In this approach, as faults have overlap to each other the starting lines of each fault (which are unique) have critical part/role. Therefore, the last five probability estimation values produced from applying detection model of start-classification provided by first layer are used directly to create each feature vector of second layer.

To implement such method, we design a buffer (first input-first output) with size of five in order to save aforementioned values. As illustrated in Figure 3.7, vectors of second layer are considered to have nine dimensions that their first five elements are filled by the values of aforementioned buffer (Buffer-Q). The procedure is started by windowing and creating numerical feature vector for the first window (FV1). Afterwards, two predicted models from first layer are applied on this feature vector.

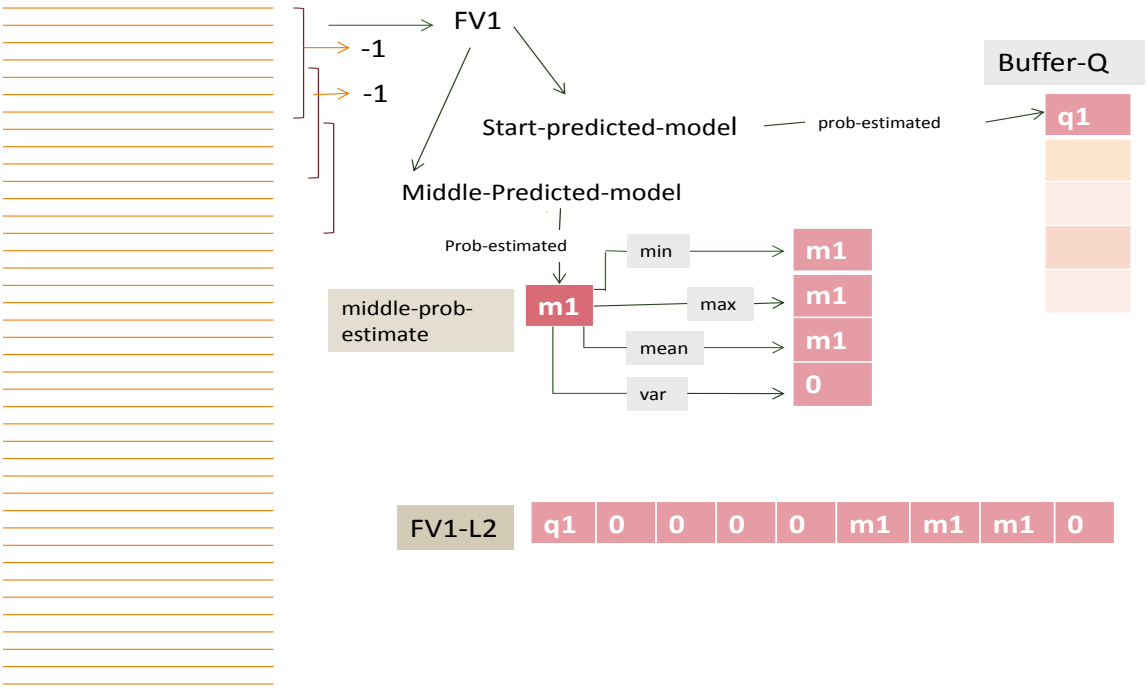


Figure 3. 7. *Creating first feature vector of second layer*

The probability estimated value obtained by applying Start-classification model (named as Start-predicted-model in Figure 3.7) is saved in the buffer while another vector (middle-prob-estimate in Figure 3.7) is used to save probability estimated value obtained by applying middle-classification model (named as Middle-predicted-model in Figure 3.7) on this feature vector (FV1). This vector is an unlimited vector in order to have the ability of saving probability estimated values of next windows. The statistical values obtained from aforementioned vector such as minimum, maximum, mean, and average value are used to create the feature vector of the second layer. The last step to generate the feature vector is to replace the first five elements of it by buffer (Buffer-Q) values and last four elements of it by statistical values of the second layer. But as it is illustrated in Figure 3.7 for the first feature vector buffer has only one value, and hence, we need to replace the other elements by zeros. Furthermore, there are three equal statistical values for the first feature vector of the second layer (FV1-L2).

For the second feature vector, the aforementioned procedure is performed on the next window. It is depicted in Figure 3.8 that probability estimated value obtained by applying Start-predicted-model on the feature vector of this window is saved in buffer while buffer has also kept the previous value. And statistical values are computed using both

probability estimated values of applying Middle-predicted-model on feature vector correspond to this window and the previous one.

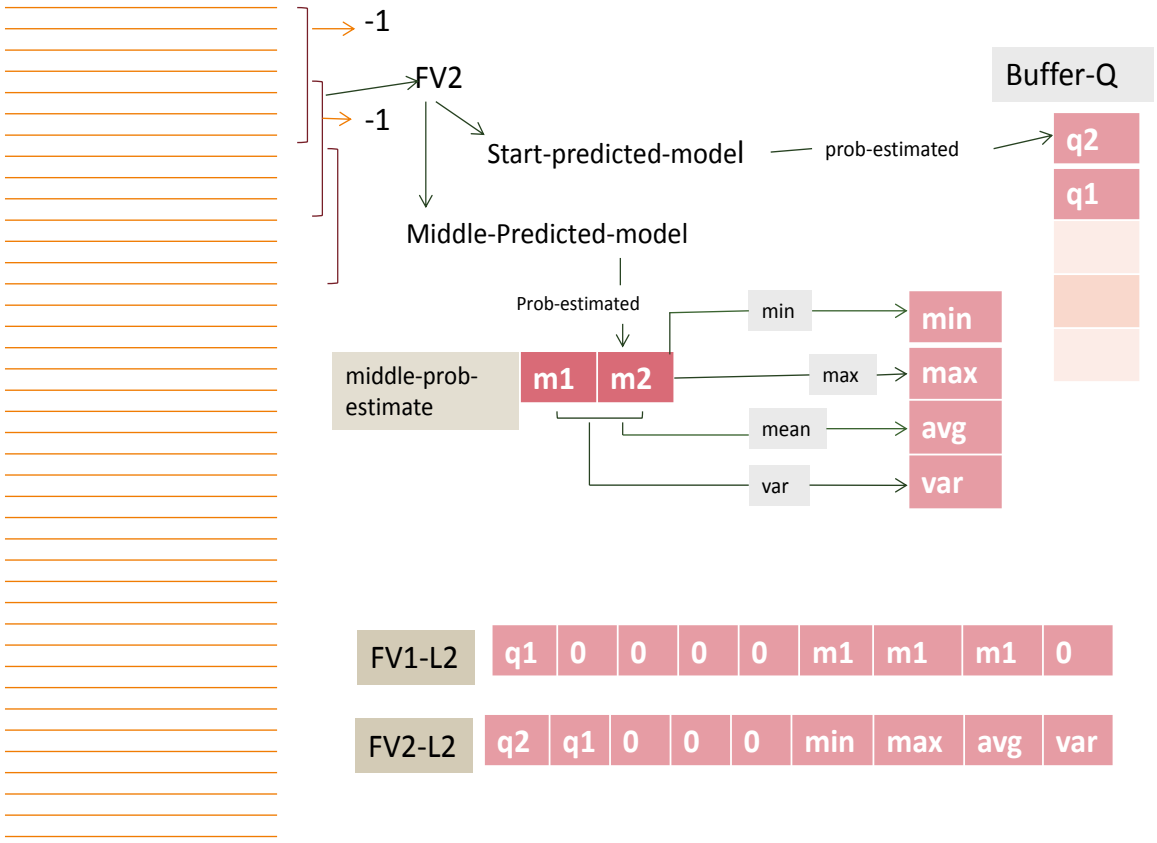


Figure 3. 8. *Creating second feature vector of second layer*

To assign label to each feature vector, we use the approach described in Sub-Section 3.1.1 under the title of labelling. It means both generating feature vector of second layer and labelling them are based on sliding window throughout the main dataset. As it is illustrated in Figure 3.9 the aforementioned procedures continue and next feature vectors are built as long as the fault logs have not been detected and all labels indicate the normal cases.

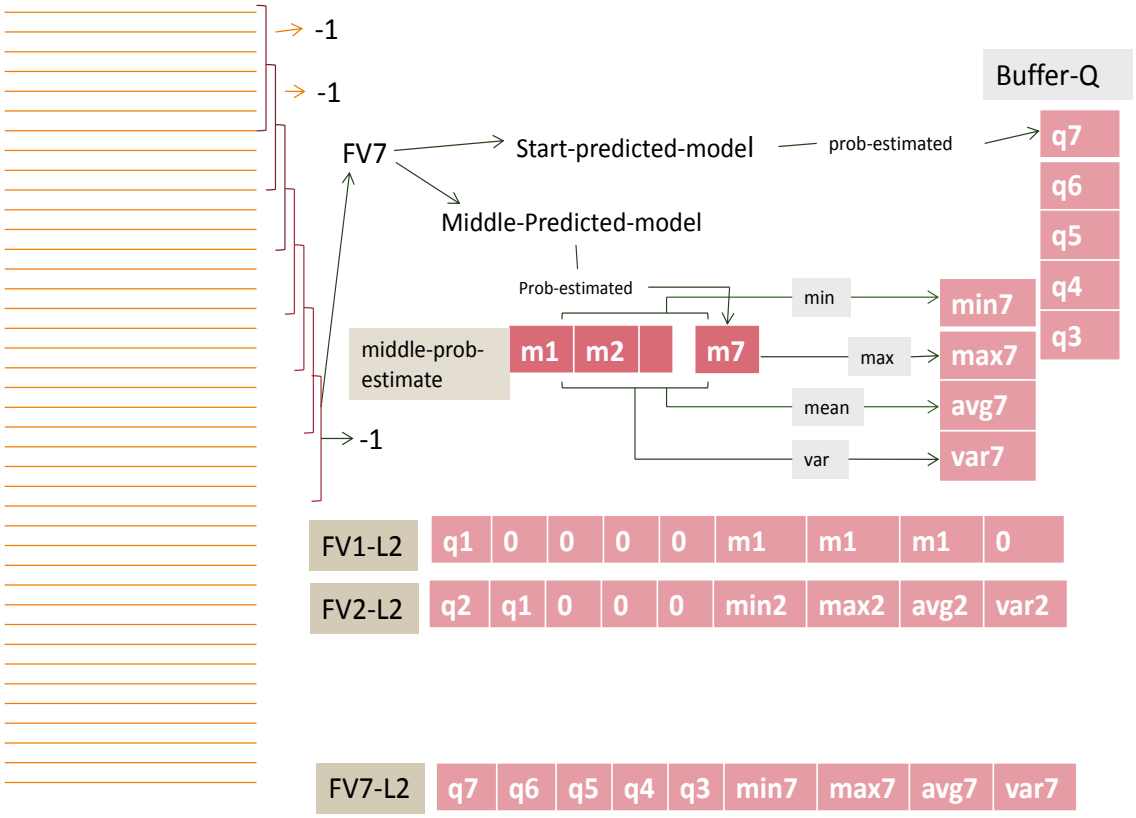


Figure 3. 9. *Feature vectors of second layer consist of last five values from start-predicted-model and statistical values from middle-predicted-model of first layer*

When the fault is detected by the label correspond to window, buffer will not update anymore and only the dimension of middle-prob-estimate increases by collecting the probability values of applying middle-Predicted-model to feature vector correspond to each window. Accordingly, as it is seen in Figure 3.10, after indicating fault logs, only the last four elements of feature vectors are updated while the first five elements are fixed. This procedure continues until the end of fault and indicating the normal logs by associated label.

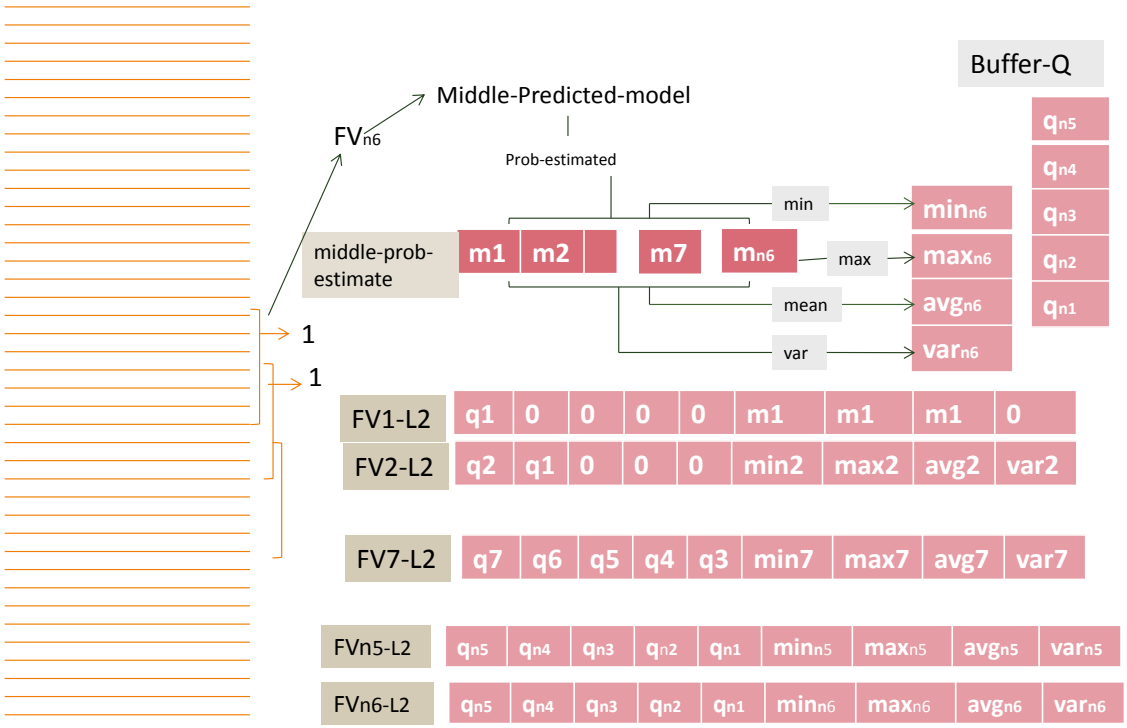


Figure 3. 10. *After detection of fault logs, only the last four elements of feature vector will be changed*

Indicating the normal situation cause both buffer and middle-prob-estimate vector, illustrated in Figure 3.11 to reset, and the procedure repeats again from the first step for the rest of data.

After collecting all feature vectors for second layer the detection model is created by applying the same learning algorithm with the exact parameters used for first layer on the training data obtained from second layer. And then this approach is evaluated by applying aforementioned model on the testing samples.

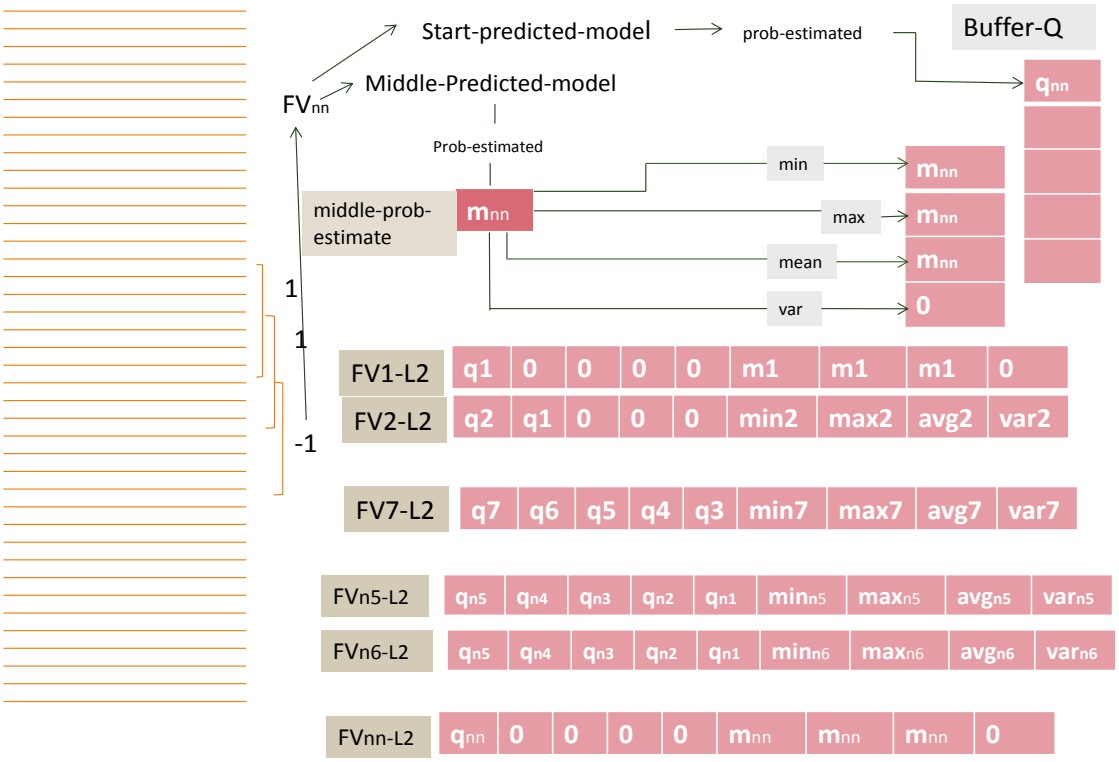


Figure 3. 11. *By ending the fault logs and indicating normal logs both buffer and middle-prob-estimate reset and procedure starts from first step*

4. RESULTS

In this chapter, we present the experimental results for our proposed fault detection technique over two dataset. We first describe the dataset characteristics and evaluation measures used for our experiments, and then present the results and discuss the performance of the detection method considering different factors.

4.1 Data

We did experiments using two different datasets, TTY and TTY-2, and compare two classifier methods considering the effects of different factors. The datasets used in this thesis are generated by TIETO using a simulator with a real cellular (3G/4G) network structure from Poland[1]. The log data contains more than traffic data. It may also include diagnosis data, system status report, error report, system performance report, and so one. The data is collected from both mobile core network and other peripheral parts, such as base stations. The faults simulated in the data are related to software, hardware, and network connection faults as well as configuration problems in the 4G network base stations. Software faults might relate to license expiring whereas hardware related issues might be network connections lost, HDD failures or access/write problems.

TTY dataset with 500330 logs is chosen to evaluate two classifier methods, single layer Perceptron and SVMs. It contains two days of network traffic data, which includes a fault with a duration time of around 16 minutes that repeats 20 times throughout the dataset. This fault is related to the base station base bandwidth failure.

TTY-2 dataset with 516691 logs contains five days of network traffic data including five different faults each one repeats 20 times throughout the dataset. These faults have different duration time around 8 milliseconds, 2 minutes, 5 minutes, 16 minutes and the most challenging fault has 36 hours duration. Thus, its 20 time repetitions cause to have a dominant fault with near 5 days duration. The faults included station base bandwidth

[1] http://beta.btsearch.pl

failure, software licensing alarm issue, a faulty configuration file, a broken cable connection, and BCCH missing alarm. TTY-2 is chosen to investigate the effect of fault duration factor on classifier method and generate a proper approach for long lasting faults.

For each experiment a time series data together with a configuration file that defines the duration time of each fault and its repetition times are available. For example, from Figure 4.1, time distance between first pair of lines (difference between two timestamps) shows the fault duration for the nineteenth repetition of the fault with id equal to 1. Second pair of lines indicates that fault with id 1 has been repeated twenty times in total. Hence, the next pair of lines represents the timestamps of the fault with id 2. This file is used to provide ground truth dataset required for supervised learning methods.

```
22.03.2014 18:30:01.148;Fault_id_1#19;Start
22.03.2014 18:52:18.048;Fault_id_1#19;Stop

23.03.2014 14:30:53.769;Fault_id_1#20;Start
23.03.2014 14:53:20.669;Fault_id_1#20;Stop

20.03.2014 19:27:01.361;Fault_id_2#1;Start
23.03.2014 07:32:02.161;Fault_id_2#1;Stop

22.03.2014 07:22:58.023;Fault_id_2#2;Start
24.03.2014 19:27:58.423;Fault_id_2#2;Stop
```

Figure 4. 1. *A part of Fault List Log illustrating two different faults, and time duration of each repetition*

4.2 Evaluation measures

As mentioned in chapter 3, in order to evaluate detection models, we need to reserve a portion of data for the test set. In our early experiments we manually split the dataset into two parts as training dataset and testing dataset. But for cases that fault logs are not equally distributed throughout the dataset or the number of one class is very small in compare to other class it is probable that one of the sets for training or testing might miss a certain class. Therefore, this method for collecting training and testing samples

cannot provide a proper approach to assess the detection model. Instead C*ross validation* [47] is a technique used to overcome this shortage and to select the optimal detection model [2] and estimate the accuracy performance of classifiers. By using this technique, feature vectors collected from entire dataset are partitioned into complementary subsets (or folds). One set is considered as testing set or validation set and used to evaluate the created model. While the other subsets called training set, are used to create the detection model. In addition, to evaluate the performance of learning algorithms we need to choose metric measures that are able to measure the best performance of learning algorithms. In literature, accuracy and error rate are two common criterion functions to assess classifier performance in order to find the best detection model. Accuracy defines the percentage of correct classifications, while error rate is the percentage of incorrect classifications. But they may not well suited for evaluating models created from imbalanced datasets when the number of abnormal logs is much less than the number of normal logs. For example in a dataset that consists of 100 logs in total and only one abnormal log, for detection results indicating all logs as normal the accuracy will be 99%. While this result represents low quality of model as it cannot detect any faults in dataset. Instead, *precision* and *recall* are two evaluation functions that focused on the number of detected faults [28]. To define these two evaluation metrics first we need to introduce the confusion matrix that represents the prediction results. Confusion matrix is a table where each cell [i,j] indicates the number of times that j was predicted when the correct label was i. Figure 4.2 shows a confusion matrix where:

- True negative (TN) corresponds to the number of normal logs correctly predicted by the learning method.
- True Positive (TP) corresponds to the number of abnormal logs correctly predicted by the learning method.
- False Negative (FN) corresponds to the number of abnormal logs wrongly predicted as normal logs by the learning method.
- False positive (FP) corresponds to the number of normal logs wrongly predicted as abnormal logs by the learning method.

		Predicted	
		Normal	Abnormal
Actual	Normal	TN	FP
	Abnormal	FN	TP

Figure 4. 2. Confusion Matrix

Thus, the diagonal elements indicate labels that were correctly predicted and the off-diagonal elements indicate errors. Precision represents the fraction of real abnormal logs from all predicted abnormal logs by a classifier method. High precision means that learning method rarely predicts normal logs as abnormal logs. Recall measures the fraction of abnormal logs that are correctly predicted by classifier method. High recall means that learning method rarely predicts abnormal logs as normal logs.

$$\text{Recall, r} = \frac{TP}{TP+FN} \tag{4.1}$$

$$\text{Precision, p} = \frac{TP}{TP+FP} \tag{4.2}$$

However, consider only one of them is not enough to evaluate a learning method. Because, for example if a model predicts all logs as abnormal logs the recall value of it will be large but its precision value will be low. On the contrary, high precision and low recall represent that all predicted logs are correct though a large number of logs are still unpredicted. Therefore, we need to trade-off between recall and precision and set the standard way that has been proposed to combine these two measures. F-score [28] is a metric that combines recall and precision to evaluate a learning model. Hence, a model with high F-score implies that both recall and precision are high enough. F-score reaches its best value at 1 and worst value at 0.

$$\text{F-score, } F_1 = \frac{2rp}{r+p} = \frac{2*TP}{2*TP+FP+FN} \tag{4.3}$$

4.3 Experimental results and Discussion

In the following subsections, the results of fault detection performance are represented. The results are divided into two sections for TTY dataset and TTY-2 datasets considering different factors and characteristics of each one.

4.3.1 TTY dataset

In this part our primary focus is to obtain the best overall classification performance regardless of the number of logs contained in each dataset and faults duration. All results in this chapter are obtained, using 10-fold cross validation to achieve as much accuracy as possible. However the results are not perfect since the samples of each fold in cross validation is selected randomly.

Performance measurements of different classifiers: As mentioned in chapter 2, regarding textual characteristics of log data linear classification is the most proper technique for detecting the faults. In this thesis two linear classification methods, single layer Perceptron and SVMs are to compare. From various implementations of SVMs, LIBSVM and LIBLINEAR are two candidate approaches to be explored in this thesis. LIBSVM [8] and LIBLINEAR [18] are both open source libraries for SVMs to help users to easily apply SVM to their applications. However LIBLINEAR focuses on large-scale problems such as text classifications as an easy-to-use tool to deal with their large dimensionality sparse characteristics [18]. When training a support vector machine several parameters can be set. One of them is the type of kernel to use. For data with the large number of features a linear kernel is preferred. LIBLINEAR is an implementation of SVMs which trains linearly without the use of kernels. From Figure 4.3, as we expect the performance of LIBLINEAR is much better than two others for all values of windowing. LIBSVM could provide high F-score for small window sizes. But it seems that by increasing the size of window that causes to decrease the number of training samples LIBSVM produces poor F-scores. And single layer perceptron provides the worst result for all window sizes among other methods.

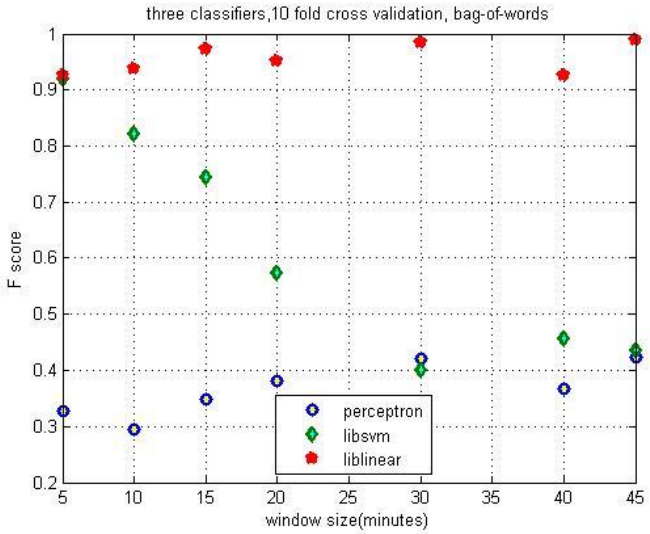
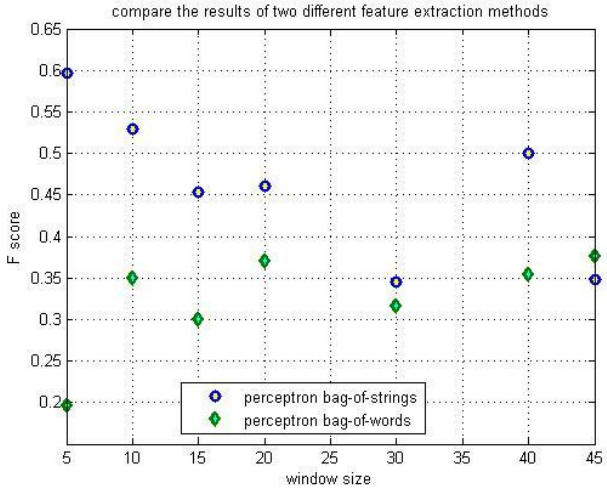


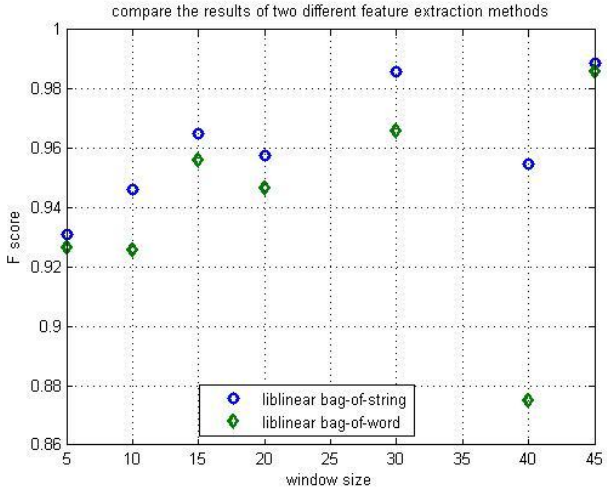
Figure 4. 3. *Performance of different classifiers*

bag-of-words vs bag-of-strings: Increasing the size of data causes to increase the number of features. As each feature corresponds to a dimension thus the dimension of feature space increases that causes to reduce the quality of detection. Our proposed approach to reduce the dimensionality problem and improve the detection performance is to use bag-of-strings instead of bag-of-words in feature generation process. Figure 4.4 show that this solution gives significant performance only on Perceptron algorithm. Because using bag-of-strings causes to reduce the sparseness problem of feature vectors, in particular for small windows that suffer this problem severely. For two other approaches, LIBLINEAR and LIBSVM, the results show that there are not significant difference between using bag-of-strings and bag-of-words. These results prove that SVM implementations are well suited methods for large sparse data.

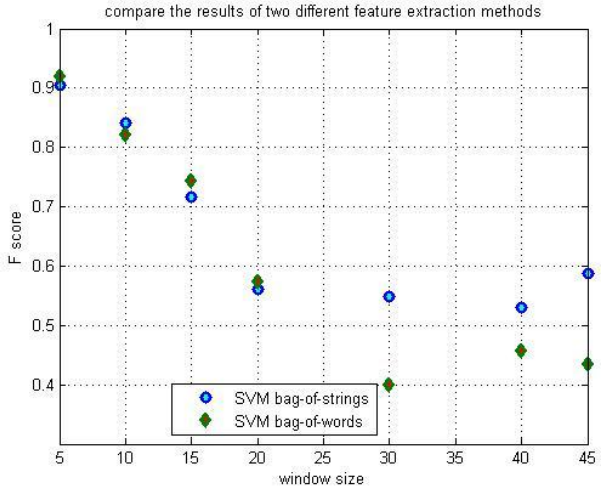
Labelling rule: The other factor that may effect on the quality of results is the rule utilized for labelling. As mentioned in Section 3.1.1, we labelled feature vectors based on the ratio of the numbers of normal logs to abnormal logs. In our experiments we examined several rules to find the best one with high quality. Figure 4.5 shows the results of two experiments for one-fourth and one-fifth rule on LIBLINEAR approach. One-fourth rule means a feature vector is labelled as fault if one-fourth or more of logs in a range of window indicate faults, otherwise it is labelled as normal.



a. Comparison of bag-of-strings and bag-of-words on Perceptron algorithm



b. Comparison of bag-of-strings and bag-of-words on Perceptron algorithm



c. Comparison of bag-of-strings and bag-of-words on LIBSVM

Figure 4. 4. *Results of bag-of-strings vs bag-of-words*

The concept of one-fifth rule is the same while only the threshold is changed to one-fifth instead of one-fourth. We expect to improve the quality of results for smaller thresholds since it causes to be more sensitive to the number of fault logs in each window. However from Figure 4.5, behavior of rules is not the same for all examined window sizes. In total the results of one-fifth rule is more satisfying for majority of window sizes. Then we prefer to continue our experiments using one-fifth rule for labelling.

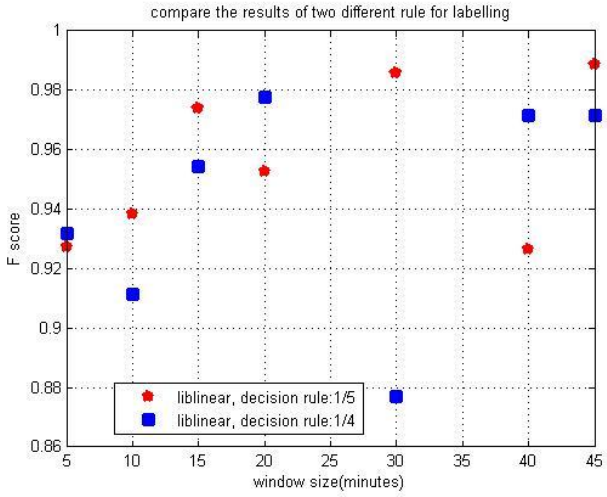


Figure 4. 5. *Results of two different rules for labelling*

4.3.2 TTY-2 dataset

TTY-2 is a dataset with five different faults including a long lasting fault. For this dataset, we consider distinct binary classification for each fault separately. In this section, the results of our proposed solution for long lasting fault are presented. Then by comparing the results of LIBLINEAR approach, , since it has the best performance among other ones, on the other faults, we attempt to find the optimal window size that can give the best performance.

Two-layer classification for long duration faults: long lasting faults cause to have abnormal dominant dataset while normal logs occur rarely. In TTY-2, this phenomenon happened since one long fault of around 36 hours (fault-2) is repeated several times throughout the dataset. This kind of dataset presents a particular challenge for learning algorithm, which causes to reduce the accuracy achieved by simple method used for TTY dataset. As mentioned in 3.2.2 to address this problem, two-layer classification is

proposed. Figure 4.6 shows the results of two different methods, single-layer and two-layer classification. It is obvious that the best performance by using two-layer classifier achieved for small window sizes that have the worst performance by using one layer classification. But, there is no significant difference for large window sizes. In addition, a drawback of this approach is that creating feature vectors for two layers and two times for first layer is significantly time consuming. In practice may be we need a trade-off between elapsed time and high quality. As we can see from Figure 4.6, by using one layer classification for windows with sizes of 40 or 45 minutes, we do not loss significant quality while we can improve the speed of implementation considerably.

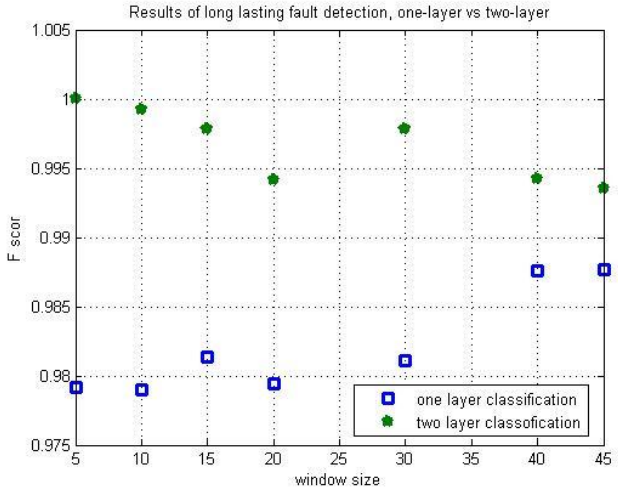


Figure 4. 6. *Performance of two-layer classification vs one layer classification*

Finding optimal window size: To find an optimal window size in order to achieve high quality of fault detection, we test the other four faults of TTY-2 using different window sizes correspond to each fault durations. With the results in Figure 4.7 to Figure 4.10, it seems that the windows smaller than fault duration are not performing well, although there is a limitation range for windows with large size. As we see, for fault 3, fault 4, and fault 5 that have small duration, their performance after a certain range of window associated to each fault drop off significantly. However, we cannot mention an exact range as optimal; it is clear that well performance is obtained for those windows that are proportional to duration of faults.

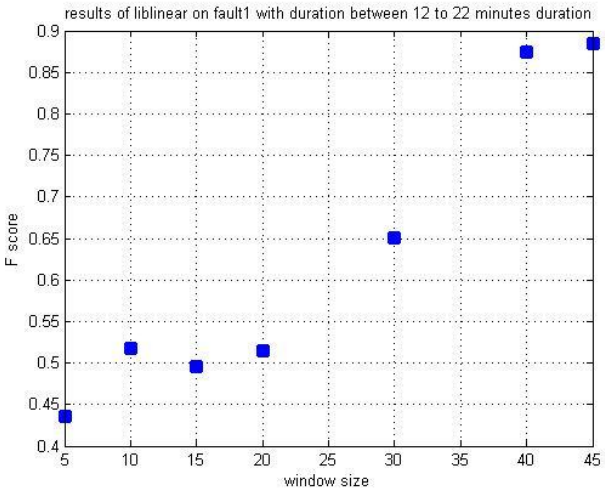


Figure 4. 7. *The results of LIBLINEAR on fault 1 with duration between 12 minutes to 22 minutes for different window sizes*

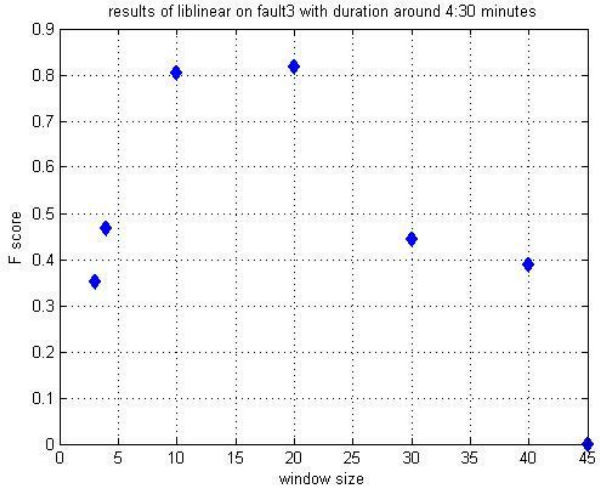


Figure 4. 8. *The results of LIBLINEAR on fault 3 with duration around 4:30 minutes for different window sizes*

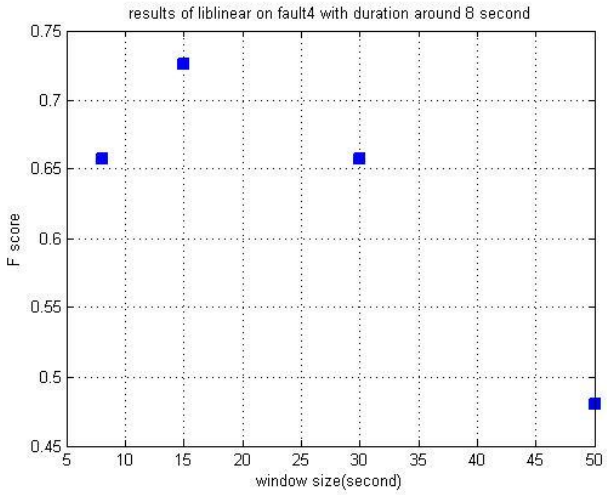


Figure 4. 9. *The results of LIBLINEAR on fault 4 with duration around 8 seconds for different window sizes*

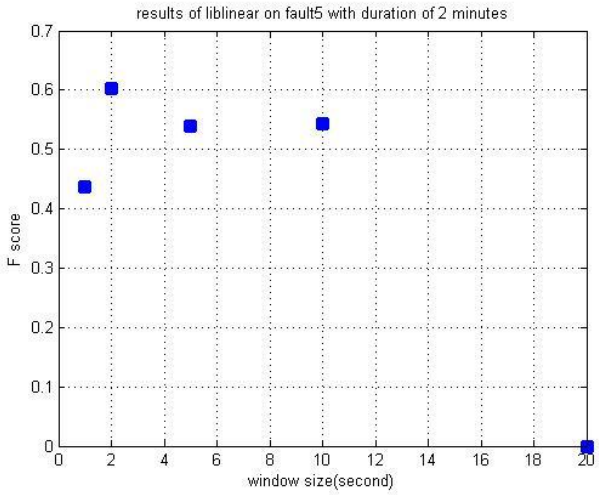


Figure 4. 10. *The results of LIBLINEAR on fault 5 with duration around 2 minutes for different window sizes*

5. CONCLUSION

The aim of this thesis was to investigate supervised machine learning approaches to detect known faults from unstructured log data simulated and provided by TIETO. To find suitable methods, we focused on textual characteristics of the log data. Our proposed approach for creating feature vectors in this thesis was inspired from methods for streaming data together with text data. Consequently, we used sliding window together with bag-of-words approaches to create feature vectors. Considering textual characteristics of network-based server-log data such as large dimensions and sparseness, we focus on single layer perceptron and SVM, as two candidate methods of linear classification methods.

The preliminary experiments on a small dataset with limited logs indicated satisfying performance of single layer perceptron. But increasing the number of logs included in a dataset causes to reduce the detection quality of Perceptron. Experimental results show that the detection quality of SVM is significantly better than single layer perceptron for datasets with large number of logs. In this thesis, we compared the implementations of LIBLINEAR and LIBSVM. As we expected, LIBLINEAR not only provided much better results than LIBSVM also its training times were shorter than LIBSVM. In order to reduce the dimension of feature space and improve the detection quality, we proposed using bag-of-strings instead of bag-of-words. But this solution was effective only for single layer perceptron and did not have considerable difference on two other approaches. This result proves high ability of SVMs to deal with high dimension sparse data.

The other factor that was investigated in this thesis was dataset with long lasting fault. Long lasting fault in a dataset causes fault class to be dominant class while normal logs occur rarely. Simple LIBLINEAR method used for dataset with limited duration fault could not provide high quality results. We proposed two-layer classification with a novel approach for creating feature vectors as a solution to improve the fault detection performance. In the first layer, we attempt to detect each occurrence of fault (or the first line of each fault occurrence) along with total duration of fault throughout the dataset

by two separate classifications. The obtained models from first layer are used for creating feature vectors for second layer. Two-layer classification demonstrates more fault detection ability than single-layer classification. Although, creating feature vectors for two layers is time consuming task. Comparison the results of two approaches (single-layer and two-layer classification) show that there are not significant differences in performance of two approaches for large window sizes. In practice, we may need a trade-off between elapsed time and high performance of detection.

A limitation of our detection approach is that this approach is unable to detect unknown faults that may appear in dataset. Although, from this thesis, we could find well suited method for extracting features and learning algorithm and investigate the effect of some factors on fault detection, more investigation is still needed to achieve better results. There are some cases that the results are against our expectations. For example, we could not find an optimal range for window for sliding window as the results of each method show different performance associated to each window size. Moreover, creating feature vectors in particular for data with large number of logs is time consuming. Besides, there are some limitations that refer to supervised learning method. For example this approach is unable to detect unknown faults that may appear in communication networks. In addition, as the approach depends on training data, any change in network and system behavior makes us update the training set and model prediction.

REFERENCES

[1] Aggarwal, Charu C., and ChengXiang Zhai. "A survey of text classification algorithms." Mining text data. Springer US, 2012. 163-222.

[2] Arlot, Sylvain, and Alain Celisse. "A survey of cross-validation procedures for model selection."Statistics surveys 4 (2010): 40-79.

[3] Axelsson, Stefan.”Intrusion detection systems: A survey and taxonomy. Vol. 99. Technical report, 2000.

[4] Bitincka, Ledion, et al. "Optimizing data analysis with a semi-structured time series database." SLAML’10: Proceedings of the 2010 workshop on Managing systems via log analysis and machine learning techniques. 2010.

[5] Bonifácio Jr, José Maurıcio, et al. "Neural networks applied in intrusion detection systems." Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on. Vol. 1. IEEE, 1998.

[6] Bowen, Theodore, et al. "Building survivable systems: An integrated approach based on intrusion detection and damage containment." DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings. Vol. 2. IEEE, 2000.

[7] Broda, Bartosz, et al. "Fextor: A feature extraction framework for natural language processing: A case study in word sense disambiguation, relation recognition and anaphora resolution." Computational Linguistics. Springer Berlin Heidelberg, 2013. 41-62.

[8] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)*2.3 (2011): 27.

[9] Ciocarlie, Gabriela F., et al. "Detecting anomalies in cellular networks using an ensemble method." CNSM. 2013.

[10] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[11] Dagan, Ido, Yael Karov, and Dan Roth. "Mistake-driven learning in text categorization." arXiv preprint cmp-lg/9706006 (1997).

[12] Dasgupta, Dipankar, and Fabio González. "An immunity-based technique to characterize intrusions in computer networks." Evolutionary Computation, IEEE Transactions on 6.3 (2002): 281-291.

[13] Debar, Hervé, et al. An experimentation workbench for intrusion detection systems. IBM TJ Watson Research Center, 1998.

[14] Denning, Dorothy E. "An intrusion-detection model." Software Engineering, IEEE Transactions on 2 (1987): 222-232.

[15] Dredze, Mark, Koby Crammer, and Fernando Pereira. "Confidence-weighted linear classification." Proceedings of the 25th international conference on Machine learning. ACM, 2008.

[16] Ensafi, Roya, et al. "Optimizing fuzzy k-means for network anomaly detection using PSO." Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on. IEEE, 2008.

[17] Eskin, Eleazar. "Anomaly detection over noisy data using learned probability distributions." (2000).

[18] Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." *The Journal of Machine Learning Research* 9 (2008): 1871-1874.

[19] Forrest, Stephanie, et al. "A sense of self for unix processes." Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on. IEEE, 1996.

[20] Fox, Kevin L., et al. "A neural network approach towards intrusion detection." (1990): 124-134.

[21] Fung, Glenn, and Olvi L. Mangasarian. "Proximal support vector machine classifiers." Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2001.

[22] Garcia-Teodoro, Pedro, et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges." computers & security 28.1 (2009): 18-28.

[23] Garcia-Teodoro, Pedro, et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges." computers & security 28.1 (2009): 18-28.

[24] Ghosh, Anup K., Aaron Schwartzbard, and Michael Schatz. "Learning Program Behavior Profiles for Intrusion Detection." Workshop on Intrusion Detection and Network Monitoring. Vol. 51462. 1999.

[25] Ghosh, Anup K., Christoph Michael, and Michael Schatz. "A real-time intrusion detection system based on learning program behavior." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2000.

[26] Ghosh, Anup K., and Aaron Schwartzbard. "A Study in Using Neural Networks for Anomaly and Misuse Detection." USENIX Security. 1999.

[27] Golab, Lukasz. "Querying sliding windows over online data streams." Current Trends in Database Technology-EDBT 2004 Workshops. Springer Berlin Heidelberg, 2005.

[28] Goutte, Cyril, and Eric Gaussier. "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation." *Advances in Information Retrieval*. Springer Berlin Heidelberg, 2005. 345-359.

[29] Habra, Naji, et al. "ASAX: Software architecture and rule-based language for universal audit trail analysis." Computer Security—ESORICS 92. Springer Berlin Heidelberg, 1992. 435-450.

[30] Hearst, Marti A., et al. "Support vector machines." *Intelligent Systems and their Applications, IEEE* 13.4 (1998): 18-28.

[31] Helman, Paul, and Gunar Liepins. "Statistical foundations of audit trail analysis for the detection of computer misuse." Software Engineering, IEEE Transactions on 19.9 (1993): 886-901.

[32] Hofmeyr, Steven A., and Stephanie Forrest. "Architecture for an artificial immune system." Evolutionary computation 8.4 (2000): 443-473.

[33] Hood, Cynthia S., and Chuanyi Ji. "Proactive network-fault detection [telecommunications]." Reliability, IEEE Transactions on 46.3 (1997): 333-341.

[34] Hu, PingZhao, and Malcolm I. Heywood. "Predicting intrusions with local linear models." Neural Networks, 2003. Proceedings of the International Joint Conference on. Vol. 3. IEEE, 2003.

[35] Hu, Wenjie, Yihua Liao, and V. Rao Vemuri. "Robust Support Vector Machines for Anomaly Detection in Computer Security." ICMLA. 2003.

[36] Idé, Tsuyoshi, and Hisashi Kashima. "Eigenspace-based anomaly detection in computer systems." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.

[37] Ilgun, Koral, Richard A. Kemmerer, and Phillip A. Porras. "State transition analysis: A rule-based intrusion detection approach." Software Engineering, IEEE Transactions on 21.3 (1995): 181-199.

[38] Jian, Guan, Liu Da-Xin, and Cui Bin-Ge. "An induction learning approach for building intrusion detection models using genetic algorithms." Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on. Vol. 5. IEEE, 2004.

[39] Jiang, Weihang, et al. "Understanding Customer Problem Troubleshooting from Storage System Logs." FAST. Vol. 9. 2009.

[40] Joachims, Thorsten. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. No. CMU-CS-96-118. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1996.

[41] Joachims, Thorsten. Text categorization with support vector machines: Learning with many relevant features. Springer Berlin Heidelberg, 1998.

[42] Kabiri, Peyman, and Ali A. Ghorbani. "Research on Intrusion Detection and Response: A Survey." IJ Network Security 1.2 (2005): 84-102.

[43] Kayacik, H. Gunes, A. Nur Zincir-Heywood, and Malcolm I. Heywood. "On the capability of an SOM based intrusion detection system." Neural Networks, 2003. Proceedings of the International Joint Conference on. Vol. 3. IEEE, 2003.

[44] Kivinen, Jyrki, Manfred K. Warmuth, and Peter Auer. "The Perceptron algorithm versus Winnow: linear versus logarithmic mistake bounds when few input variables are relevant." Artificial Intelligence 97.1 (1997): 325-343.

[45] Kline, Jeff, et al. "Traffic anomaly detection at fine time scales with bayes nets."Internet Monitoring and Protection, 2008. ICIMP'08. The Third International Conference on. IEEE, 2008.

[46] Ko, Calvin, Manfred Ruschitzka, and Karl Levitt. "Execution monitoring of security-critical programs in distributed systems: A specification-based approach." Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on. IEEE, 1997.

[47] Kohavi, Ron. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *IJCAI*. Vol. 14. No. 2. 1995.

[48] Kruegel, Christopher, et al. "Bayesian event classification for intrusion detection." Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003.

[49] Kumar, Sandeep, and Eugene H. Spafford. "A software architecture to support misuse intrusion detection." (1995).

[50] Kumpulainen, Pekka, and Kimmo Hätönen. "Anomaly detection algorithm test bench for mobile network management." Tampere University of Technology(2008).

[51] Laguna, Javier Ortiz, Angel García Olaya, and Daniel Borrajo. "A dynamic sliding window approach for activity recognition." User Modeling, Adaption and Personalization. Springer Berlin Heidelberg, 2011. 219-230.

[52] Lane, Terran, and Carla E. Brodley. "Temporal sequence learning and data reduction for anomaly detection." ACM Transactions on Information and System Security (TISSEC) 2.3 (1999): 295-331.

[53] Lee, Wenke, and Salvatore J. Stolfo. "Data mining approaches for intrusion detection." Usenix Security. 1998.

[54] Lee, Wenke, et al. "Real time data mining-based intrusion detection." DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings. Vol. 1. IEEE, 2001.

[55] Liao, Yihua, and V. Rao Vemuri. "Use of k-nearest neighbor classifier for intrusion detection." Computers & Security 21.5 (2002): 439-448.

[56] Martineau, Justin, et al. "Improving binary classification on text problems using differential word features." Proceedings of the 18th ACM conference on Information and knowledge management. ACM, 2009.

[57] Mukkamala, Srinivas, Guadalupe Janoski, and Andrew Sung. "Intrusion detection using neural networks and support vector machines." Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on. Vol. 2. IEEE, 2002.

[58] Náther, Peter. "N-gram based Text Categorization." Lomonosov Moscow State Univ (2005).

[59] Ng, Hwee Tou, Wei Boon Goh, and Kok Leong Low. "Feature selection, perceptron learning, and a usability case study for text categorization." ACM SIGIR Forum. Vol. 31. No. SI. ACM, 1997.

[60] Nowak, Eric, Frédéric Jurie, and Bill Triggs. "Sampling strategies for bag-of-features image classification." *Computer Vision–ECCV 2006*. Springer Berlin Heidelberg, 2006. 490-503.

[61] Papapetrou, Odysseas, Minos Garofalakis, and Antonios Deligiannakis. "Sketch-based querying of distributed sliding-window data streams."Proceedings of the VLDB Endowment 5.10 (2012): 992-1003.

[62] Papineni, Kishore. "Why inverse document frequency?." Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies. Association for Computational Linguistics, 2001.

[63] Patcha, Animesh, and Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends." Computer Networks 51.12 (2007): 3448-3470.

[64] Perdisci, Roberto, Guofei Gu, and Wenke Lee. "Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems."Data Mining, 2006. ICDM'06. Sixth International Conference on. IEEE, 2006.

[65] Radovanović, Miloš, and Mirjana Ivanović. "Text mining: Approaches and applications." Novi Sad J. Math 38.3 (2008): 227-234.

[66] Ramadas, Manikantan, Shawn Ostermann, and Brett Tjaden. "Detecting anomalous network traffic with self-organizing maps." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2003.

[67] Salem, Malek Ben, and Salvatore J. Stolfo. "Detecting masqueraders: A comparison of one-class bag-of-words user behavior modeling techniques."Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 1.1 (2010): 3-13.

[68] Schütze, Hinrich, David A. Hull, and Jan O. Pedersen. "A comparison of classifiers and document representations for the routing problem." Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1995.

[69] Sebastiani, Fabrizio. "Machine learning in automated text categorization." ACM computing surveys (CSUR) 34.1 (2002): 1-47.

[70] Sekar, R., et al. "Specification-based anomaly detection: a new approach for detecting network intrusions." Proceedings of the 9th ACM conference on Computer and communications security. ACM, 2002.

[71] Shao, Jun. "Linear model selection by cross-validation." *Journal of the American statistical Association* 88.422 (1993): 486-494.

[72] Silva, Catarina, and Bernardete Ribeiro. "The importance of stop word removal on recall values in text categorization." Neural Networks, 2003. Proceedings of the International Joint Conference on. Vol. 3. IEEE, 2003.

[73] Sinka, Mark P., and David W. Corne. "A large benchmark dataset for web document clustering." Soft Computing Systems: Design, Management and Applica tions 87 (2002): 881-890.

[74] Sipola, Tuomo, Antti Juvonen, and Joel Lehtonen. "Anomaly detection from network logs using diffusion maps." Engineering Applications of Neural Networks. Springer Berlin Heidelberg, 2011. 172-181.

[75] Snapp, Steven R., et al. "DIDS (distributed intrusion detection system)-motivation, architecture, and an early prototype." Proceedings of the 14th national computer security conference. Vol. 1. 1991.

[76] Stearley, John. "Towards informatic analysis of syslogs." Cluster Computing, 2004 IEEE International Conference on. IEEE, 2004.

[77] Tamaru, Ann, et al. A real-time intrusion-detection expert system (IDES). SRI International, Computer Science Laboratory, 1992.

[78] Tan, Kymie MC, and Roy A. Maxion. "Determining the operational limits of an anomaly-based intrusion detector." Selected Areas in Communications, IEEE Journal on 21.1 (2003): 96-110.

[79] Tang, Jiliang, Salem Alelyani, and Huan Liu. "Feature Selection for Classification: A Review."

[80] Tseng, Chin-Yang, et al. "A specification-based intrusion detection system for AODV." Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks. ACM, 2003.

[81] Vapnik, Vladimir. The nature of statistical learning theory. springer, 2000.

[82] Vigna, Giovanni, Steven T. Eckmann, and Richard A. Kemmerer. "The STAT tool suite." DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings. Vol. 2. IEEE, 2000.

[83] Xu, Wei, et al. "Detecting large-scale system problems by mining console logs."Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, 2009.

[84] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." ICML. Vol. 97. 1997.

[85] Zhang, Tong, and Frank J. Oles. "Text categorization based on regularized linear classification methods." Information retrieval 4.1 (2001): 5-31.

[86] Zhang, Yongguang, and Wenke Lee. "Intrusion detection in wireless ad-hoc net works." Proceedings of the 6th annual international conference on Mobile compu ting and networking. ACM, 2000.