

Anomaly Detection On A Real-Time Server Using
A Data Mining Algorithm

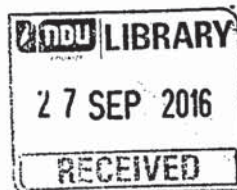
By

Georges Chaaya

Department of Computer Science,
Notre Dame University

A dissertation submitted to Notre Dame University,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

June 2015

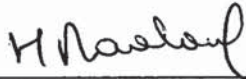


Anomaly Detection On A Real-Time Server Using A Data Mining Algorithm

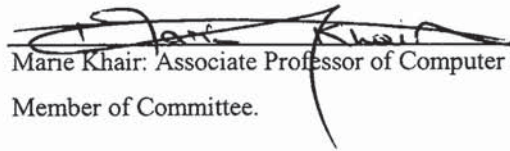
By

Georges Chaaya

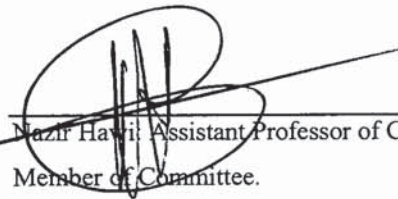
Approved by:



Hoda Maalouf: Associate Professor of Computer Science
Advisor.



Marie Khair: Associate Professor of Computer Science
Member of Committee.



Nazir Hawil: Assistant Professor of Computer Science
Member of Committee.

Date of Thesis Defense: June 26, 2015

Abstract

Anomaly detection is the process of finding outlying record from a given data set. This problem has been of increasing importance due to the increase in the size of the data and the need to efficiently extract those outlying records that can have important indications in real-life problems. Anomaly detection is applied in many different sectors.

There are many approaches to solve the anomaly detection problem. However, those that are more widely applicable are unsupervised approaches as they do not require labeled data.

The aim of this thesis is to study a well-known anomaly detection technique on a specific application, and to find some ways to tune and optimize it in order to have better precision and results.

The server chosen in this thesis is the Short Message Service Centre server, which is used in the telecommunications field to handle and store messages (SMSs) so that they can be properly delivered to the appropriate destination. This server was studied in details to decide which data should be taken into consideration. Once done, a script was written to gather all these data, which went through a cleaning phase (preprocessing) and then through a labeling process after being deeply analyzed and divided into many subsets.

After extensive research, the decision tree algorithm was chosen to be implemented, and it was applied to the labeled data obtained. The original tree that was constructed gave a precision of 98.82%. After that, different types of tuning were performed to increase the precision, which reached 99.38% with an effective accuracy of 99.98% (relative to our case). Our approach proved that the application on this type of servers is efficient and it leads to very good results, which can also be improved in future studies.

Acknowledgements

I would like to thank Dr. Hoda Maalouf for her supervision and patience throughout the research and writing of this thesis. I would also like to thank Dr. Marie Kheir for her help and the data mining concepts she supplied.

Many thanks also go to all the NDU faculty members who have been part of my educational path and to all my friends who supported me when I needed it the most.

And the biggest “Thank you” goes to God, who has been with me all along the way, and who without him I couldn’t have reached the stage I am at now.

Dedication

To my family, who offered me unlimited support and guidance throughout my life and motivation during this thesis, and to my close friends who were next to me during this period.

Table of Contents

Chapter 1 Introduction and Problem Definition.....	1
1.1 Introduction to the General Problem.....	1
1.2 Problem Definition.....	1
1.3 Research Objectives.....	2
1.4 Approach and Main Results.....	3
1.5 Thesis Organization.....	3
Chapter 2 Anomaly Detection.....	5
2.1 Definitions.....	5
2.1.1 Anomaly.....	5
2.1.2 Anomaly Detection.....	6
2.2 Importance and Applications.....	6
2.3 Challenges.....	8
2.4 Aspects.....	9
2.4.1 Input.....	9
2.4.2 Types.....	10
2.4.2.1 Point Anomalies.....	10
2.4.2.2 Contextual Anomalies.....	10
2.4.2.3 Collective Anomalies.....	11
2.4.3 Labels.....	11
2.4.4 Output.....	12
2.5 Related Work.....	12
Chapter 3 Anomaly Detection Techniques.....	14
3.1 Basic Approaches to Anomaly Detection.....	14
3.1.1 Supervised Anomaly Detection.....	14
3.1.2 Semi-Supervised Anomaly Detection.....	15
3.1.3 Unsupervised Anomaly Detection.....	15
3.2 Classification Algorithms.....	15
3.3 Advantages and Disadvantages of Classification Data Mining Techniques.....	17
3.4 Decision Tree Techniques.....	18
3.5 The C4.5 Tree Construction Algorithm.....	19
3.6 Overfitting, Pruning and Cross Validation.....	22

3.7 Classifying an Unseen Instance.....	23
3.8 The Confusion Matrix	23
Chapter 4 Anomaly Detection in Real-Time Servers.....	26
4.1 Introduction	26
4.2 System Context.....	26
4.2.1 Distributed Architecture	26
4.2.2 External Connections.....	28
4.2.2.1 The Mobile Network	28
4.2.2.2 The External Short Messaging Entities (ESMEs)	28
4.2.2.3 The Billing System	29
4.2.2.4 The Real-time Charging Gateway (RTCG).....	29
4.2.2.5 The Rules Engine Unit (REU).....	29
4.3 SMSC Software.....	30
4.4 Possible Anomalies in a Distributed Architecture.....	31
4.5 Processes and Features	32
4.5.1 Synchronization Processes	32
4.5.2 Mobile Number Portability.....	33
4.5.3 Distribution Lists.....	33
4.5.4 IP Failover	33
4.5.5 The Web Interface	34
4.5.6 Statistics.....	34
4.5.7 Address Translation.....	34
4.5.8 Lawful Interception	35
4.5.9 Tracking.....	35
4.5.10 Auto-Signature	35
Chapter 5 The Implementation.....	37
5.1 Server Related Studies.....	38
5.2 Important Issues to Address	39
5.3 Initial Data Selection	41
5.4 The Script Used for Data Selection.....	43
5.5 Cleaning the Data or Data Preprocessing.....	44
5.5.1 Handling Duplicates	44

5.5.2 Handling Missing Values.....	45
5.6 Adding Anomalous Records.....	45
5.7 Differences in Attribute Types.....	46
5.8 Correlations Between Attributes.....	47
5.9 Seasonality.....	49
5.10 Data Labeling.....	51
5.11 WEKA Software.....	52
5.12 Data Manipulation.....	53
5.12.1 Choosing the Typical Number of Records.....	53
5.12.1.1 The Training Time.....	54
5.12.1.2 The Tree Size.....	55
5.12.1.3 The Accuracy.....	55
5.12.2 Between Cross-Validation and Percentage Split.....	56
5.12.3 Applying Tuning Concepts.....	58
5.12.3.1 Combining Multiple Attributes into One.....	58
5.12.3.2 Replacing Attribute Values by One of Two Values.....	59
5.12.3.3 Applying the Concept of Ranges.....	61
5.13 Statistics.....	63
5.13.1 Three-Class Model.....	63
5.13.2 Two-Class Model.....	64
5.14 Transforming the Model into a Script.....	66
5.15 The Alarm System.....	66
Chapter 6 Conclusion and Future Work.....	67
6.1 Conclusion.....	67
6.2 Main Contribution of this Thesis.....	67
6.3 Possible Extensions and Future Work.....	68
Appendix A Script to Gather the Data.....	70
Appendix B Sample Model Source Code.....	75
Bibliography.....	77

List of Figures

Figure 2-1 Anomalies in a 2-Dimensional Data Set.....	5
Figure 3-1 Multi-Class VS. One-Class Anomaly Detection	16
Figure 3-2 Pseudocode of the C4.5 algorithm.....	20
Figure 3-3 The Confusion Matrix.....	24
Figure 4-1 Distributed Architecture	27
Figure 4-2 System Context.....	30
Figure 5-1 Handling Duplicates	45
Figure 5-2 Relation Between SMSC Process and Counter	47
Figure 5-3 Relation Between Messages Table and MySQL Size.....	48
Figure 5-4 Average CPU Usage For 6 Days	50
Figure 5-5 WEKA GUI	53
Figure 5-6 Training Time Vs. Number of Records	54
Figure 5-7 Tree Size Vs. Number of Records	55
Figure 5-8 Accuracy Vs. Number of Records	56
Figure 5-9 Statistics - Case 1	63
Figure 5-10 Detailed Accuracy - Case 1	63
Figure 5-11 Confusion Matrix - Case 1	64
Figure 5-12 Statistics - Case 2.....	65
Figure 5-13 Detailed Accuracy - Case 2	65
Figure 5-14 Confusion Matrix - Case 2.....	65

List of Tables

Table 3-1 Advantages and Disadvantages of Classification-Based Techniques.....	17
Table 3-2 Confusion Matrix Formulas.....	25
Table 5-1 Cases with Different Training Data File Size.....	57
Table 5-2 Modification Cases.....	60
Table 5-3 Tuning Cases.....	62

List of Abbreviations

AIM	Application Interface Module
CDR	Customer Data Record
ESME	External Short Message Entity
GMSC	Gateway Mobile-services Switching Center
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HLR	Home Location Register
HTML	Hyper Text Markup Language
MO	Mobile Originated
MPU	Message Processing Unit
MNP	Mobile Number Portability
MSC	Mobile Switching Center
MSISDN	Mobile Station ISDN Number
MT	Mobile Terminated
OAM	Operation, Administration and Maintenance
PLMN	Public Land Mobile Network
SIU	Signaling Interface Unit
SMPP	Short Message Peer-to-Peer
SMS	Short Message Service
SS7	Signaling System Number 7
STP	Signaling Transfer Points
TCP/IP	Transmission Control Protocol / Internet Protocol
TLR	Transaction Log Record

Chapter 1

Introduction and Problem Definition

1.1 Introduction to the General Problem

Outliers were considered in the past as noisy data in statistics. Recently, their detection has become a very important problem which is undergoing much research in different fields and application domains. The purpose is to be able to catch these abnormal events or behaviors that might cause harm to the related application in order to prevent any negative effect.

Many specific techniques have been developed to certain applications, while others are more general. Some are very critical to the point that they are being researched in strict confidentiality.

As a specific application, some studies tackled the problem of anomaly detection on servers. When dealing with some types of servers, it is an essential task to be able to accurately identify the anomalies. Efforts were done to find algorithms and techniques that catch these anomalies with high accuracy and to apply them to the servers.

In current real-time applications, it is important to find new ways to discover anomalies and improve the results and accuracies of these techniques.

1.2 Problem Definition

Different algorithms were used to solve the anomaly detection problem. The server, consisting of hardware and software, is connected to many other entities, and on which many processes are running. This server is constantly receiving data, generating data, and its normal functioning is essential to the service being provided.

During some periods, this server might have some issues that can affect the service: hardware failures, software errors, loss of connections with the external entities, and many others. The purpose of all server-related research is to be able to prevent any loss of service, and this is why some analysis for this specific server needs to be done to catch any malfunctioning before it causes disastrous results.

In order to have accurate analysis, a specific server that is being currently implemented was considered in this research (SMSC server), and also the data used is real data collected from this server.

The server at hand was not being monitored before. Engineers were relying on basic monitoring tools and they would generally wait for an error to occur in order to investigate the reason behind it and try to solve it. The goal of this thesis is to be able to monitor the server continuously and fire an alarm whenever something suspicious happens.

1.3 Research Objectives

The main objective of this thesis is to detect anomalies in real world servers. It is essential to understand the concept of anomaly detection, how it is applied, and in which fields. Then, different techniques should be gone through, those that are used to detect anomalies, in order to choose a specific server and decide which technique should be implemented on this server. After that, the goal would be to perform some tuning relative to the application at hand, and to customize it based on the use cases encountered. The objective is to obtain a very accurate model that will be able to catch any anomaly that occurs on the system.

1.4 Approach and Main Results

First, we start by studying anomaly detection in details, focusing on the importance of detecting anomalies ahead of time, the applications where this concept is used, the challenges that are faced when tackling this problem, and the different aspects of it.

Second, we go through the basic approaches used to detect anomalies, the domain of each approach (where it is applied), and the server on which the study is done; we show its context, architecture, role, connections, features, and everything we need to understand in order to decide what to include in our monitoring process, what is of high importance, and what can be ignored at some point.

Then we have to decide which algorithm should be used in this thesis and explain it along with all important concepts related to it. This algorithm should be applied to the server, and tests have to be conducted to improve the accuracy and find the best customized model.

Finally, the model will be implemented and tested on the system.

The algorithm used was the decision tree algorithm, and it lead to a very high accuracy when applied on the SMSC (99.83%).

1.5 Thesis Organization

This thesis is organized into six chapters. Chapter 2 discusses the concept of anomaly and anomaly detection, different studies that were done on this subject, the advantages and disadvantages, the difficulties and challenges, and the most important points to consider when working with anomaly detection. Chapter 3 introduces the basic approaches to anomaly detection, and then describes the data mining algorithm used. It shows the different aspects of the decision tree, how it is built, and how to understand the meaning of the results it gives. Chapter 4 presents the SMSC server on which the study was conducted in order to explain what should be monitored. Chapter 5 shows our main contribution to this research field, the

progress of the study and the customization of the decision tree algorithm along with the experiments and tests that were done. Finally, Chapter 6 summarizes the main results and contributions of the thesis and provides some of its possible extensions.

Chapter 2

Anomaly Detection

2.1 Definitions

2.1.1 Anomaly

An “anomaly” or an “anomalous object” is an object (point) that is sensibly different from other objects (points). In statistics, an “outlier” is an observation that is numerically distant from the rest of the data [1].

Anomalies are patterns in data that do not conform to a well-defined notion of normal behavior. Figure 1 illustrates anomalies in a simple 2-dimensional data set. The data has two normal regions, N1 and N2. Points that are sufficiently far away from the regions (e.g., points O1 and O2, and points in region O3) are anomalies [2].

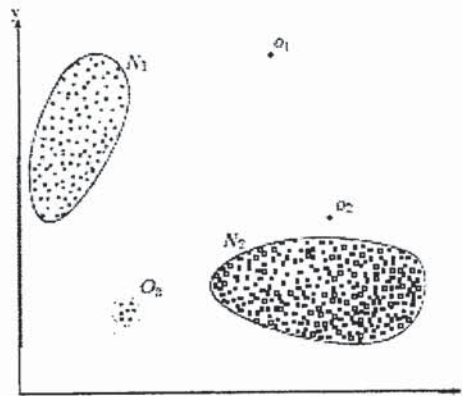


Figure 2-1 Anomalies in a 2-Dimensional Data Set

2.1.2 Anomaly Detection

Anomaly detection is the process of finding outlying record from a given data set [3]. It refers to the problem of finding patterns in data that do not conform to expected behavior. These non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, faults, damage, aberrations, surprise, peculiarities or contaminants in different application domains [2].

Detecting outliers or anomalies in data has been studied in the statistics community as early as in the 19th century [4].

Outliers arise due to different reasons such as mechanical faults, changes in system behavior, fraudulent behavior, human error and instrument error.

Anomaly Detection combines machine learning, statistical analysis and human knowledge from your domain experts to analyze streams of machine data, detect events in the stream and provide alerts on those events, allowing you to remediate issues before they affect business services [5].

2.2 Importance and Applications

This problem has been of increasing importance due to the increase in the size of the data and the need to efficiently extract those outlying records.

The anomalies in data often translate to meaningful information in many application domains, and this information can help in detecting problems or new strange phenomena, discovering unusual behavior in the data, and can sometimes be really critical and can lead to extremely disastrous results that cause damage to the related application.

Some examples include unauthorized access of a system, credit card theft or the diagnosis of a disease [3]. For example, if an anomalous traffic pattern is encountered in a computer network, this could be related to a hacked computer that is sending out sensitive data to an unauthorized destination [6]. If a certain anomaly is detected in an MRI image, it may

indicate that some malignant tumors are present [7]. Anomalies in credit card transaction data can indicate that the credit card was stolen, or that an individual might be subject to an identity theft [8]. In military surveillance, an unusual region present in a satellite image in an enemy area can indicate enemy troop movement [9]. In safety critical environments, the presence of an outlier indicates abnormal running conditions that can cause significant performance degradation [2]. Anomalous readings from a space craft sensor could signify a fault in some component of the space craft, such as an engine rotation defect or a flow problem in a pipeline.

The anomaly detection has been useful in different fields, and has been applied in a variety of applications, among which the following:

- Industrial Damage Detection (Fault Detection in Mechanical Units, Structural Defect Detection...)
- Image Processing
- Novel Topic Detection in Text Data
- Sensor Networks
- Fraud Detection (Credit Card Fraud Detection, Mobile Phone Fraud Detection, Insurance Claim Fraud Detection, Insider Trading Detection...)
- Intrusion Detection (host-based intrusion and network intrusion)
- Ecosystem Disturbances (trying to predict events like hurricanes and floods)
- Medical and Public Health Anomaly Detection (using unusual symptoms or test result to indicate potential health problems).
- Speech Recognition
- Novelty Detection in Robot Behavior
- Traffic Monitoring
- Faults in Web Applications
- Astronomical Data Anomaly Detection

Online anomaly detection is an important step in data center management, requiring light-weight techniques that provide sufficient accuracy for subsequent diagnosis and management actions [10].

Detecting anomalies, as mentioned previously, can be extremely valuable. For instance, one can start talking about being able to detect warnings about a possible future failure in a certain system, or to notice a variation in some health indicators, which might indicate the possible presence of a certain disease, or to encounter some unusual patterns of shopping cart failures on an ecommerce website.

Furthermore, anomalies don't always indicate a failure, and therefore they are not always bad. For example, if a change is detected in a consumer buying habits, this could give the business an opportunity to discover a new trend.

The number of machines and computers that generate data is highly increasing, which means that the opportunity for detecting anomalies is growing rapidly [11].

2.3 Challenges

As mentioned earlier, an anomaly is a pattern that does not conform to the expected normal behavior. A straightforward approach would be to define a region that represents normal behavior and to consider any instance in the data that does not belong to this normal region as an anomaly. But this approach is very challenging due to many factors:

- It is nearly impossible to define a normal region that covers all normal behaviors, and the boundary between normal and anomalous behavior is often not precise. Therefore, an instance classified as anomalous and that lies close to the boundary can be actually normal.
- In many cases where the anomaly is a result of malicious actions, the attackers usually try to hide their action by making the anomalous observations appear like normal, which makes it difficult to define normal behavior.
- The systems are constantly changing over time, software is being always updated, even the patterns of the human interaction change. Therefore, normal behavior often evolves, and a

normal behavior today can be considered as anomalous sometime in the future. This is why, in order to have effective anomaly detection, the systems have to learn continuously.

- The notion of anomaly varies with the application. Each application has its own constraints and therefore leads to a specific problem formulation to detect anomalies.
- In most of the cases, the data is not labeled or very hard to label.
- The data usually contains noise that might be similar to the anomalies, to the point that it cannot be easily distinguished from the anomalies and consequently cannot be removed.
- It's not very efficient to just wait for a certain metric to be largely out of bounds in order to confirm that an anomaly is present. All systems should be able to detect any small change in patterns, even if this change is not easily detected, in order to predict the possibility of the anomaly occurrence before it actually happens [11].

Because of the challenges mentioned above, the anomaly detection problem is not easy to solve.

Most of the existing techniques focus on a specific formulation which is affected by the nature of the data, whether it is labeled or not, the type of anomaly to be detected [2].

2.4 Aspects

2.4.1 Input

Whatever the anomaly detection technique is, it is very important to know the nature of the input data. In general, it is a collection of data instances (also referred as object, record, point, vector, pattern, event, case, sample, observation, entity) [12].

The data instances are described using a set of attributes (also referred to as variable, characteristic, feature, field, dimension), which can be binary, categorical or continuous.

The data instance can be univariate (consisting of one attribute) or multivariate (consisting of many attributes). For multivariate data instances, the attributes don't have to be of the same type.

The nature of the attributes can be essential in determining which anomaly detection technique to choose. If statistical techniques are to be used, these techniques vary according to whether the attributes are continuous or categorical.

In most of the existing anomaly detection techniques, it is assumed that no relationship exists between the data instances. But in general, these instances can be related to each other. For example, in sequence data, the data instances are linearly ordered (like the case in protein sequences), in special data (like vehicular traffic data) the instances are related to their neighboring instances. These relationships among data instances can be very relevant for anomaly detection.

2.4.2 Types

The purpose of anomaly detection is to find data patterns that do not conform to the expected behavior. Based on the nature, context, behavior or cardinality, anomalies can be generally classified in three categories [13]:

2.4.2.1 Point Anomalies

This is the simplest type of anomaly. It refers to an instance of the data that is anomalous with respect to the rest of the data. This type occurs in most of the applications, and it was addressed by a good amount of research.

2.4.2.2 Contextual Anomalies

Also known as conditional anomalies. This type is defined for a data instance that is anomalous in a specific context (and not otherwise). In general, the structure of the dataset induces the notion of context.

Two sets of attributes are used to determine if a data instance belongs to this type of anomaly:

Contextual attributes: they determine the context of the instance. For example, in some cases time can be a contextual attribute that helps to specify the position of the instance.

Behavioral attributes: they determine the non-contextual characteristics of an instance. For example, in a spatial data sets that describes the average salary of employees in a certain country, the amount of salary in a specific region can be defined as a behavioral attribute. In order to determine an anomalous behavior, the values of behavioral attributes have to be taken within a specific context. A data instance might be an anomaly in a certain context, but it could be considered normal in a different context.

2.4.2.3 Collective Anomalies

They refer to collections of related data instances that are anomalous with respect to the entire set of data. The individual data instances might not be considered as anomalies by themselves, but when they occur together as a collection they form an anomalous event. Point anomalies can occur in any data set, whereas collective anomalies occur only in data sets where the data instances are related. On the other hand, contextual anomalies occur where context attributes are available in the data. Point or collective anomalies can also be contextual anomalies if analyzed with respect to a context.

2.4.3 Labels

A label is a tag associated with a data instance to denote if that instance is normal or anomalous.

It is usually very expensive and hard to get labeled data that cover all scenarios and types of behavior. Experts do the labeling process manually, and it requires a great effort and a long time. Also, obtaining a labeled set of anomalous instances is much harder than getting labels for normal instances, because the probability of occurrence of an anomalous behavior is usually very low. Adding to it that the anomalous events are often dynamic, in the sense that new anomalies that have no labeled data sets can arise, especially when it comes to events that are rare and catastrophic.

Based on the availability of labels, anomaly detection techniques can operate in three modes that will be discussed in details later: supervised, semi-supervised and unsupervised.

2.4.4 Output

In all of the anomaly detection techniques, the anomalies can be reported in two ways: by means of scores or labels:

An anomaly score is assigned to each instance in the test data depending on the degree of anomaly. Therefore, analysts will have a ranked output and can choose to analyze only the top anomalies based on a threshold they specify.

On the other hand, binary labels are assigned to the test data instances as normal or anomalous.

2.5 Related Work

Many surveys, books, and articles tackled the problem of anomaly detection: Hodge and Austin [14] wrote a detailed survey of the anomaly detection techniques that are developed in machine learning and statistical domains. Agyemang et al. [15] presented a more general review on anomaly detection techniques for numeric and symbolic data. Novelty detection techniques using neural networks and statistical approaches were presented by Markou and Singh [16]. Patcha and Park [17] studied the anomaly detection techniques used for intrusion detection in computer networks, and Lazarevic et al. [18] presented an evaluation of these techniques. Forrest et al. [19], Snyder [20] and Dasgupta and Nino [21] reviewed anomaly detection techniques that were specifically developed for system call intrusion detection.

To deal with the anomalies types mentioned above, different techniques have been proposed over the decades. For the point anomaly type, distance-based approaches were mainly used, but their effectiveness depends on many factors, among which the type and

dimensionality of data and the anomaly score used. For the contextual anomalies, in addition to distance-based approaches, density-based approaches were also used. Here also the same factors previously mentioned play a vital role. Collective anomaly is mainly handled by density-based approaches, but in the identification of this type of anomalies, new factors affect the decision such as compactness and single linkage effects [22].

Looking at the problem independently from the anomaly type, the approaches that are more widely applicable are unsupervised approaches since they do not require labeled data. Even though many techniques exist, the strength and weaknesses of each of them are still unfolding [3].

Some of these techniques include statistical methods, distance and model based approaches, and profiling methods [23][24][25]. Another part of anomaly detection would be generating artificial counter-examples from the unknown class [26][27][28]. Its advantage is that a standard generic classifier can be trained to separate the regular class from the anomalous class. It has a great performance, but the sampling of the anomalous class might be hard, especially when dealing with high dimensionality. This problem was addressed by applying Active Learning [26], and Feature Bagging, using a decision tree as classifier. In [29], an extension for standard decision tree algorithms for anomaly detection that can deal with continuous and symbolic features is proposed. With this method, there is no need to generate artificial samples of the missing class into the training set, but instead, a parametric distribution of the outlier class can be used when determining the split points which are more accurate in this case, and the training is faster due to the presence of fewer samples.

Chapter 3

Anomaly Detection Techniques

3.1 Basic Approaches to Anomaly Detection

“Machine learning is a technique by which a computer “learns” from a set of data given to it, and is then able to predict the result of new data similar to the training data. The machine learning algorithm is meant to identify patterns based on different characteristics or “features” and then make predictions on new, unclassified data based on the patterns “learned” earlier.” [30].

As mentioned previously, there are three main techniques to tackle the problem of anomaly detection:

3.1.1 Supervised Anomaly Detection

These techniques assume that a training data set that has labeled instances for normal and anomalous classes is available. A predictive model is built for both classes and any new data instance will be compared to these models to determine which class it belongs to. The issues encountered in these techniques are that the anomalous instances in the training data set are much less than the normal ones, and it is very challenging to obtain accurate labels especially for the anomalous class. Therefore, some techniques tried to inject artificial anomalies in the normal data set [31][32][33].

3.1.2 Semi-Supervised Anomaly Detection

These techniques assume that the training data contains labels for the normal class only. They are much more applicable than the supervised techniques, since they do not require labels for the anomaly class. The approach used here is to build a model for the normal class, and based on this model, identify anomalies in the test data.

3.1.3 Unsupervised Anomaly Detection

They don't require training data, that is why they are the most applicable techniques. They assume that the normal instances are much more than the anomalous instances in the test data (this assumption has to be true, or else the techniques will cause a high rate of false alarms).

3.2 Classification Algorithms

The purpose of classification algorithms is to find a model for the "class" attribute using a training set (which is a set of records in the database). So given a set of records, the values of the attributes will be examined, along with the relation of these attributes with the class of the tuple, in order to extract a model that will be used to decide on the class of newly added tuples.

Classification based anomaly detection techniques go through two phases: The training phase learns a classifier using the labeled training data, and the testing phase uses the classifier to classify test instances as normal or anomalous. These techniques can be grouped into two categories: multi-class techniques which assume that the training data set includes labeled instances that belong to more than one normal class, and one-class techniques where all training instances have one class label. Figure 3-1 shows the difference between multi-class and one-class anomaly detection.

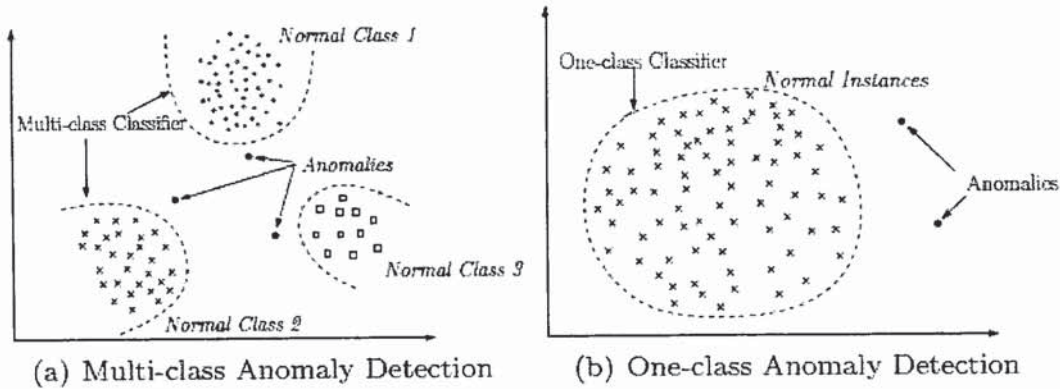


Figure 3-1 Multi-Class VS. One-Class Anomaly Detection

In anomaly detection, different techniques that use different classification algorithms to build classifiers can be used. In the following, some of these techniques are mentioned, without going in depth in each of them, since they are not applied in this thesis.

- Neural Networks: They have been applied to both one-class and multi-class. Various neural networks were used, like Multi Layered Perceptrons, Neural Trees, Auto-associative Networks, Adaptive Resonance Theory Based, Radial Basis Function Based, Hopfield Networks, Oscillatory Networks.
- Bayesian Networks: have been mainly used for multi-class anomalies.
- Support Vector Machines.
- Rule Based.

The entire classification process involves a lot of intermediate steps such as data preprocessing, clustering, feature construction, feature selection, classification, regression and finally visualization.

3.3 Advantages and Disadvantages of Classification Data Mining Techniques

When there exists training instances for both the normal and the anomalous class, it is an easy task for classification based anomaly detection techniques to accurately learn classifiers to detect anomalies, since they have enough information about normal as well as anomalous behavior. Some techniques (such as SVMs) are limited to continuous data, but other techniques (like decision trees) can be applied to categorical data also.

One of the essential problems here is that the anomalies are not often labeled. In these cases, the semi-supervised and unsupervised classification techniques should be used, knowing that these techniques are effective only when the anomalies are very far from the decision boundary.

One of the challenges presented here is when choosing between generalization and rejection. Generalization refers to the fact that unseen test instances are being accepted as normal, whereas rejection refers to considering an unseen test instance as anomalous. This issue highly depends on the threshold used on the decision confidence.

In the following Table 3-1, the main advantages and disadvantages of classification based techniques are mentioned:

Table 3-1 Advantages and Disadvantages of Classification-Based Techniques

Advantages	Disadvantages
Use powerful algorithms that can distinguish between instances belonging to different classes.	Often impossible to have accurate labels for all normal classes
Testing phase is fast	Choosing between generalization and rejection
Can be applied to both continuous and categorical data	

In addition to classification based techniques, other techniques were used to solve the anomaly detection problem, among which the nearest neighbor based, clustering based, statistical based, and graph based.

3.4 Decision Tree Techniques

From the research done, it was clear that no solution can be considered the best for all classification problems. Every classification problem has its own best solution in terms of the algorithm, the tool to use and the type and size of data required for learning. Usually, the larger the training data set is, the more accurate the prediction results are.

A decision tree is a very powerful way of presenting rules. Classification can be performed using simple computations and can be done for both continuous and categorical variables, as mentioned previously. Decision trees have many advantages when compared to other classification methods, that is why they are more suitable for outlier detection. They are among the easiest machine learning techniques to implement, to debug, to customize and also the fastest in terms of learning and classification. Usually, the speed and simplicity lead to a compromise on the accuracy of the prediction, but as it will be shown later, a well-tested and customized tree learning system can give very accurate predictions, and would perform even better than a neural network or a SVM.

Their structure can be easily interpreted, and they are also less susceptible to the curse of dimensionality [34].

Until now, many research has been done on decision trees, and it has been used in many applications such as electricity energy consumption [35], prediction of breast cancer [36], accident frequency [37].

One of the mostly used decision tree algorithm is the C4.5 which is used in this thesis, and which will be explained in details in the following.

3.5 The C4.5 Tree Construction Algorithm

C4.5 is a technique that produces at the same time a decision tree and rule sets. This model is easy to understand, from the fact that the rules derived from the tree have a clear interpretation [38]. Most of the modern decision trees are based on the C4.5 algorithm. As a general view, the algorithm chooses the best attribute(s) to split the tree and makes it a decision node. It repeats this process recursively for each child and stops when all the instances have the same target attribute value, or when there are no more attributes or instances.

Before explaining in details how the algorithm works, it is important to understand that each tuple or record is formed of values corresponding to a collection of attributes, and a class. The values of the attributes may be discrete (having a definite number of possible values) or continuous (can have any value belonging to a certain interval), whereas the class can only have discrete values.

The C4.5 algorithm constructs a tree that consists of nodes and leaves. At each node, an attribute is selected and a test on that attribute is performed. For each possible result of this test, a child node is created:

- If the attribute is discrete, it can have x possible outcomes, so the corresponding node will have x child nodes, each of them belonging to a specific possible value of the attribute.
- If the attribute is continuous, the node will usually have two possible outcomes, based on a comparison relative to a certain threshold, so one of the child nodes will correspond to a value of the attribute which is greater than or equal to a certain reference value (threshold), and the other child node will correspond to a value less than the threshold.

The class values will be at the leaves of the tree, and these are the values predicted by the tree. A certain performance error (classification error) will be present at the leaves, which corresponds to giving a class value to a certain tuple that actually belongs to another class.

Starting from a training set, the C4.5 algorithm constructs a decision tree with a “divide and conquer” strategy: At the beginning, the root is created and all the training set is associated with this node. At each node, the C4.5 algorithm shown in Figure 3-2 is executed [39]:

```
FormTree(T)
(1) ComputeClassFrequency(T);
(2) if OneClass or FewCases
    return a leaf;
    create a decision node N;
(3) ForEach Attribute A
    ComputeGain (A);
(4) N.test= AttributeWithBestGain;
(5) if N.test is continuous
    find Threshold;
(6) ForEach T' in the splitting of T
(7) if T' is Empty
    Child of N is a leaf
    else
(8) Child of N = FormTree (T');
(9) ComputeErrors of N;
    return N
```

Figure 3-2 Pseudocode of the C4.5 algorithm

T is the set of all the cases present at the node.

The class frequency $freq(C_j, T)$ is computed, which corresponds to the cases in T where the class is C_j .

If all the cases in T belong to the same class C_j , or if a few number of cases (usually less than a certain threshold) exists, the node becomes a leaf, and it is associated to the class C_j .

If the cases in T belong to two or more classes, then the information gain of each attribute is calculated, and the attribute with the highest information gain will be selected as the test at that particular node.

If the selected attribute is continuous, the threshold should be calculated.

Now whether the attribute is continuous or discrete, there will be s children corresponding to the different outputs of the test (clearly, s will be equal to 2 if the attribute is continuous).

For each subset T_i , if T_i is empty, the child node will be a leaf, and it will be assigned the most frequent class in the parent node. If T_i is not empty, the same operations will be applied recursively to T_i , adding to it the cases of T that have an unknown value of the selected attribute.

The final step would be to calculate the classification error of each node which is the sum of the errors of the child nodes. If this sum is greater than the error of classifying all cases in T as belonging to the most frequent class in T , then the node is set to be a leaf and all subtrees are removed.

To be able to calculate the information gain of an attribute, the entropy must be calculated first, according to the following formula:

$$Entropy(T) = - \sum_{j=1}^{N(Classes)} \frac{freq(C_j, T)}{|T|} * \log_2 \left(\frac{freq(C_j, T)}{|T|} \right)$$

Therefore, the information gain is:

$$Gain = Entropy(T) - \sum_{i=1}^s \frac{|T_i|}{|T|} * Entropy(T_i)$$

Other than selecting the information gain, C4.5 relies on another option, which is the information gain ratio, given by the following:

$$Split(T) = - \sum_{i=1}^s \frac{|T_i|}{|T|} * \log_2 \left(P \frac{|T_i|}{|T|} \right)$$

If the selected attribute is continuous, the cases in T that have a known value for the attribute will be ordered using a quicksort algorithm, and it is assumed that the ordered values are called V_1 to V_n .

For each i between 1 and n , This parameter is computed:

$$v = \frac{(v_i + v_{i+1})}{2}$$

This value of v will enable us to make a splitting, and to put the tuples in two categories: one that contains tuples with the value of the attribute that is less than v , and the other one with the tuples where the value of the attribute is greater than v .

Now many cases are available, and each one will give a splitting in two categories. So in order to determine which one among them is the best splitting, the information gain for each value v will be calculated by considering the splitting already explained. The value v for which the gain is maximum will be considered to be the local threshold.

3.6 Overfitting, Pruning and Cross Validation

The “Overfitting” phenomenon can be explained as follows: If the training set was too large or if the examples taken for learning are very rare occurrences, the tree model obtained may create branches for some features that are too specific and that do not make a difference to the target feature. In this case, the performance of the model that fits the training data will increase, but whenever the class of an unknown instance needs to be predicted the performance will clearly decrease.

To understand “overfitting” more easily, it is useful to consider that the training data is divided into two parts. There is data that provides information and that leads to building a precise model, and data that is simply considered as noise, and that needs to be removed from the training set.

To remove this noise, the so-called “cross-validation” is used, followed by “pruning” the tree.

In cross-validation, the training data is partitioned into two sets: the training set and the validation set. The system is trained on the training set and after the training is done, the

validation set is used to make the predictions. Each record in the validation set now has an actual value and a predicted value. The algorithm compares these two values in order to provide statistics among which the precision, recall, and most importantly the “root mean squared error rate”, because based on it the system decides the pruning strategy for the tree. In order to avoid any bias in the information, many rounds of cross validation are performed, and in each round a new set is taken for training. This will enable the algorithm to give an accurate measure of the precision. Usually, 10-fold cross validation is the most commonly used strategy.

The accuracy of the prediction depends on the size of the tree:

If the tree is very large, it might give a bad prediction accuracy on generalized data.

If the tree is very small, it might have not captured some very important features in the training data, which also leads to incorrect classifications.

Pruning a decision tree consists of removing nodes or making sub trees into leaf nodes. The best way in pruning is to let the tree grow until each node has at least few instances and then prune all the nodes that do not provide newer information.

3.7 Classifying an Unseen Instance

Before building the tree, the entropy is calculated for each feature. During this process, the minimum and maximum values for every feature become known to the system. The decision tree can be thought of as a set of rules. When an instance has to be classified, the system parses the rules and the decision is made according to the conditions that satisfy the instance. If no rule is satisfied, the decision tree has a final rule that sets the target variable to a default value.

3.8 The Confusion Matrix

A confusion matrix, commonly named “contingency table”, contains information about the actual and the predicted classifications done by a classification system. The performance of these systems is evaluated using the data in the matrix. The following Figure

3-3 shows the confusion matrix for a two class classifier that applies to the subject of this thesis:

		Predicted	
		Normal	Anomalous
Actual	Normal	a	b
	Anomalous	c	d

Figure 3-3 The Confusion Matrix

- a is the number of **correct** predictions that an instance is **normal**.
- b is the number of **incorrect** predictions that an instance is **anomalous**.
- c is the number of **incorrect** predictions that an instance **normal**.
- d is the number of **correct** predictions that an instance is **anomalous**.

In the following, the most important metrics that were defined for the 2 class matrix are mentioned, along with the formula used to calculate their values:

- The *accuracy* (AC) is the proportion of the total number of predictions that were correct.
- The *recall* or *true positive rate* (TP) is the proportion of positive cases that were correctly identified, i.e. the proportion of instances classified as class x , among all instances which truly have class x .
- The *false positive rate* (FP) is the proportion of negatives cases that were incorrectly classified as positive, or the proportion of instances classified as class x but not belonging to x , among all instances which are not of class x .
- The *true negative rate* (TN) is defined as the proportion of negatives cases that were classified correctly.

- The *false negative rate* (FN) is the proportion of positives cases that were incorrectly classified as negative.
- The *precision* (P) is the proportion of the predicted positive cases that were correct. i.e. the proportion of the instances that truly have class x among all those which were classified as class x .

If the number of negative cases is much greater than the number of positive cases, the accuracy determined by the first equation may not give a precise value. In this case, the geometric mean and the F-Measure (which is a combined measure for precision and recall) would give a more adequate result. In the F-measure formula, β can take values from 0 to infinity and is used to control the weight assigned to TP and P . All the formulas related to the above mentioned metrics can be found in Table 3-2.

Table 3-2 Confusion Matrix Formulas

$AC = \frac{a+d}{a+b+c+d}$	$TP = \frac{d}{c+d}$	$FP = \frac{b}{a+b}$
$TN = \frac{a}{a+b}$	$FN = \frac{c}{c+d}$	$P = \frac{d}{b+d}$
$g - mean_1 = \sqrt{TP * P}$	$g - mean_2 = \sqrt{TP * TN}$	$F = \frac{(\beta^2 + 1) * P * TP}{\beta^2 * P + TP}$

Chapter 4

Anomaly Detection in Real-Time Servers

4.1 Introduction

The aim of this thesis is to study anomaly detection in real-time servers. In order to have an accurate study, a Short Message Service Centre server (SMSC) was considered.

As a first step, and in order to be able to decide what to monitor on the existing system, a study was done to fully understand the SMSC role, its functionalities, and the different connections to other external entities. In this chapter, only the concepts needed in this study will be presented.

4.2 System Context

The SMSC is a server that handles and stores messages so that they can be properly delivered to the appropriate destination.

Whenever a certain mobile phone (referred to by “A-party”) sends an SMS to another mobile (B-party), the SMS goes through different elements in the network and eventually reaches this server before being sent again to the network.

4.2.1 Distributed Architecture

The SMSC works according to a distributed architecture. Logically, the server might encounter a failure at a certain point of time. The service therefore will automatically stop, which is unwanted because the messages of the customers will fail to be delivered during the failure period. Therefore, the SMSC consists of at least two servers that are active at the same time. This will provide high availability (if one SMSC fails, the others will handle all the

traffic normally), and load sharing (instead of having one SMSC handling all the messages, they will be split between the available active SMSCs).

The Distributed SMSC consists in our case of two Message Processing Units (MPUs) and two Signaling Interface Units (SIUs). One of them is designated as primary and the second as secondary.

The MPUs synchronize files between each other, handle normal short messages (incoming and outgoing), store waiting messages, and deal with charging.

The SIUs distribute the incoming messages among the MPUs, send the outgoing messages to the appropriate network elements.

In our case, the study is limited to only one SMSC and specifically to one MPU (Of course the same concept can be applied to all SMSCs belonging to the same cluster). The distributed architecture of the SMSC is presented in Figure 4-1.

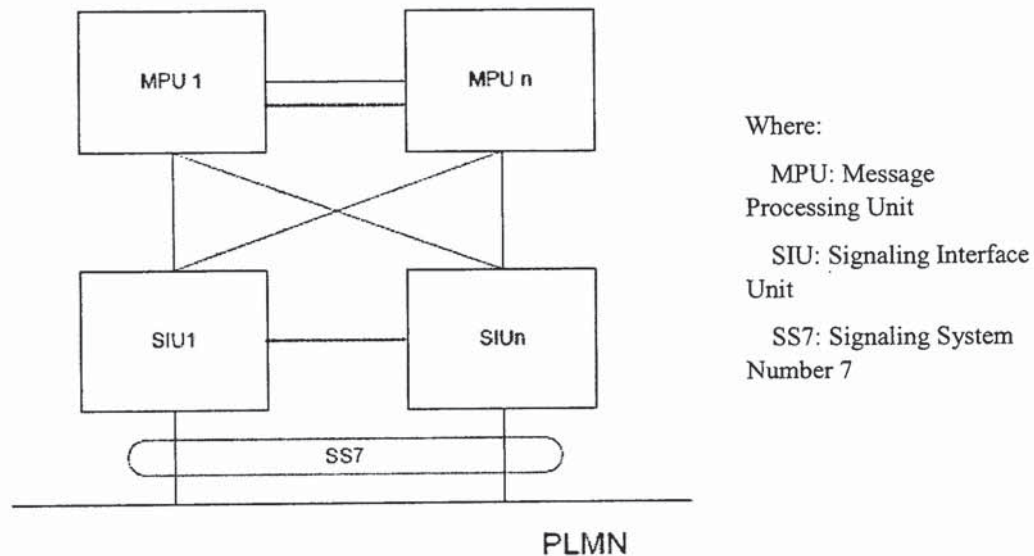


Figure 4-1 Distributed Architecture

4.2.2 External Connections

The SMSC connects to different external entities. The most essential ones are explained in the following:

4.2.2.1 The Mobile Network

Through the Signaling Interface Unit (SIU) which implements the Gateway Mobile-services Switching Centre (GMSC). The SMSC communicates with the network using IS-41 Mobile Application Part (MAP) protocol over SS7. It can communicate directly with Mobile Switching Centres (MSC) and Home Location Registers (HLR), or indirectly through Signaling Transfer Points (STP).

4.2.2.2 The External Short Messaging Entities (ESMEs)

These can send/receive messages to/from mobile subscribers. In our case, it does this using the SMPP protocol (Short Message Peer to Peer). Each ESME interface uses a transmitter to transmit messages to the SMSC and a receiver to receive messages from the SMSC. It can also use a transceiver, which performs these two actions simultaneously. Processes on the SMSC known as Application Interface Modules (AIMs) handle each of these connections.

The inter-process communication between the SMSC and the interface servers is done through the queues. Each ESME connection has its own queue. These queues have to be monitored in order to see how many messages are stuck inside them at a certain point of time. The increase in the number of these messages can indicate the slowness of the execution or that a certain process stopped unexpectedly.

In a distributed architecture, i.e. if more than one SMSC exist in the same cluster, an Application Router (MR) is needed to distribute the traffic among these SMSCs. In this case (which is similar to the case studied here), the AIMs are connected directly

to this router. Therefore, the increase in the number of stored messages in a certain queue can indicate that the related connection between the AIM and the MR was dropped for a certain reason.

4.2.2.3 The Billing System

The SMSC has a billing interface, which assists the operator in the task of charging the subscribers. This interface includes a specific format for billing and provisioning records that are sent to the billing system.

The SMSC writes in a log file a record for each event (MO, alert, result of MT attempt, etc...) which contains different fields related to the short message. These log files are referred to as TLRs. The operator does not need all the fields, so there exists a customized billing process which takes only specific fields and then generates the CDRs. Obviously, this process is very critical since it's related to the charging part, and any anomaly here can cause money loss.

4.2.2.4 The Real-time Charging Gateway (RTCG)

It connects to the Prepaid System to handle prepaid billing. It interfaces with different vendors such as Siemens' IN, Ericsson's IN... The prepaid subscribers list can be stored on the SMSC. So for each prepaid message that needs to be charged, the SMSC can add a record to an external prepaid queue, and then the charging will then be done based on a customized application between the SMSC and the operator's billing system. The prepaid process is also one of the most important processes.

4.2.2.5 The Rules Engine Unit (REU)

Upon receiving a message, the SMSC sends it to the REU in order to decide what to do with it. Based on predefined rules, it returns the corresponding action back to the SMSC (whether to accept, reject, or modify something in the message parameters).

This can help prevent spamming on the system. It can for example reject all messages that are being sent from a specific A-party, or messages that have a specific content or a text that follows a certain format.

The SMSC also connects to the Operation, Administration, and Maintenance (OAM) System. The connections to the SMSC that were mentioned above are shown in Figure 4-2.

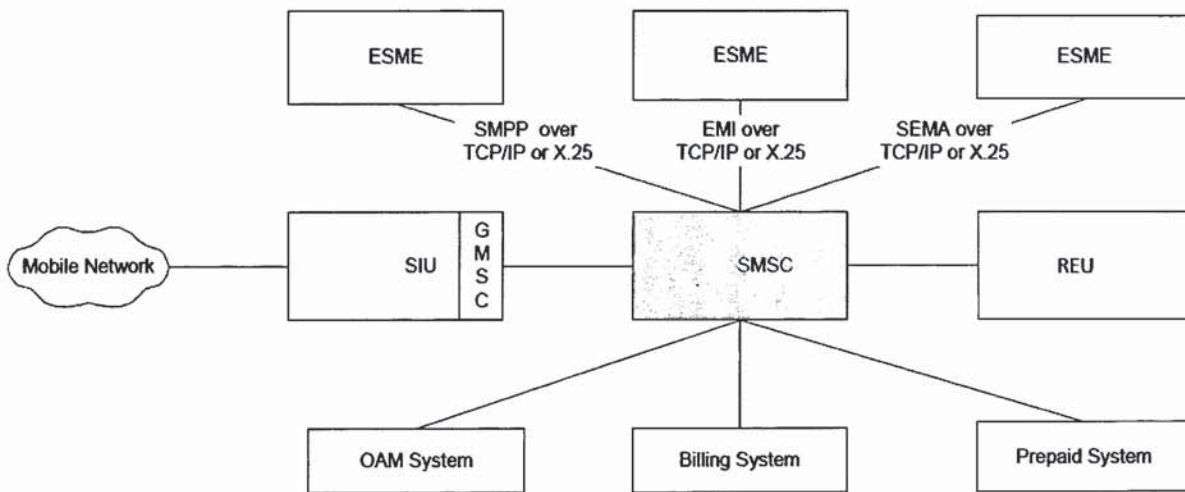


Figure 4-2 System Context

4.3 SMSC Software

The software that runs on the SMSC server consists of the following:

- A Linux operating system
- A MySQL database system (for all database related tasks)
- The SMSC application software (which is already installed on the server)

The SMSC has different modes of operation. It's not essential here to explain each mode in details, but it's important to know that certain messages are only delivered in one attempt, so if they fail they will be automatically discarded. Other messages can be stored in queues for later retries, while others are stored in the database. This concept will also help in deciding what data is important for the monitoring of the system.

4.4 Possible Anomalies in a Distributed Architecture

As explained in a previous section, the SMSC cluster is formed of two SMSC servers running simultaneously to ensure availability in case one of them goes down.

Ideally, every server in a cluster runs uniform configurations: identical resources, versions of OS, software, and data. But in reality, that is rarely the case. Systems are plagued by a number of inconsistencies that increase over time. This divergence of servers in the data centre causes a large number of problems in IT operations, from minor disruptions to large-scale outages.

“We were rolling out a major new release of our software when we found that the search results on one site turned up different results than on another. We checked code line by line and re-deployed, but the problem persisted. After scouring through the file system, we found that one of the search libraries was a different version. We reinstalled, we rebuilt the search index, and we restarted the service. Everything went well, but it had been a frustrating and time-consuming exercise. That inconsistency must have been in the search indexer for years, but until we pushed out code that used new functionality we wouldn't have known the problem existed.” - Jeremy Hutchings, Technical Director, MetroLyrics.com (a division of CBS Interactive)

These anomalies can be classified into two types:

- Environment anomalies: which result from the lack of uniformity between servers. Even if the systems are identically provisioned, differences might happen during the course of normal operations.
- Behavior anomalies: they are related to inconsistent behavior that does not conform to expectations. Systems are always provisioned identically and they receive similar traffic, therefore they should behave identically. Environment drift can cause behavior anomalies, but even without drift, a server can exhibit a behavior unlike the others in the cluster, due to a line of bad code, network congestion, hardware hiccups, or a security breach.

These concepts can also be included when studying servers in a cluster.

4.5 Processes and Features

In this section, the main processes that run on the SMSC server are described:

4.5.1 Synchronization Processes

Each MPU is considered as an SMSC node. It holds a lot of data, including the profiles of the subscribers, configurations, services, messages, and many others.

The secondary MPU connects to the primary MPU for synchronization purposes: The primary MPU receives the updates of the data, writes them into log files, and sends this information to the secondary MPU, so that all the information will be synchronized between the two MPUs.

Different synchronization processes run on both MPUs, to maintain the connection between them and to transfer the needed data. The tasks they perform include detecting the most up-to-date database, synchronizing the database, history files and configuration files, assigning roles to the MPUs, detecting failures, and starting some of the services.

4.5.2 Mobile Number Portability

The Mobile Number Portability (MNP) is the ability for mobile subscribers to keep their number when they change from one network operator to another one. For GSM subscribers, this means that the subscribers are given a new IMSI while retaining their MSISDN. Without MNP, a certain range of fixed numbers is allocated to each operator. The SMSC uses source address ranges to filter unallowed MO messages, so if an A-party is not in the defined range then it's not allowed to send messages using this SMSC. With the introduction of MNP, the SMSC relies on IMSI checking to filter its own MO subscribers. The MNP process should always be running on the SMSC so that the messages incoming from these subscribers are handled and delivered to their corresponding destinations.

4.5.3 Distribution Lists

Distribution lists is a feature that allows an operator or an authorized user to send a message to a large number of subscribers. It has a process running on the server.

4.5.4 IP Failover

The SMSC server has automatic IP failover. Each SMSC machine has a virtual IP address in addition to the real IP address. This virtual address is shared between the primary and secondary nodes. Whenever the primary node fails, the secondary node takes over the virtual IP address.

In order for this to happen, the presence of "Heartbeat" is required. Heartbeat is a daemon that provides cluster infrastructure (communication and membership) services to its clients. By this, the MPU node is able to know about the presence (or disappearance) of peer processes on the other MPU(s) and to easily exchange the messages with them.

4.5.5 The Web Interface

The SMSC is administered and configured through a user friendly Graphical User Interface (GUI). This web interface allows to easily change the parameters defined in the configuration file on the SMSC.

In a distributed architecture, the SMSC web administration pages offer a centralized management of all the nodes: each modification made from the web interface, will be distributed to all nodes. And this requires that apache is installed and configured on the servers, and it has to be always running. Otherwise, the users won't be able to use the web interface, which is run from a local PC in an Internet browser window.

4.5.6 Statistics

Statistics counters are present on the server; they provide an idea about the status and performance of the system.

The SMSC provides the administrator with different counters that offer statistics regarding the total usage of all the external connections with the SMSC as well as statistics per connection.

These counters have to be also checked deeply, since they give very important information regarding the running processes, the connections, and the traffic.

4.5.7 Address Translation

The address translation is applied for every message handled by the SMSC. Two translation files are used: one to translate the originating and destination addresses into an international format needed for profiling and internal searching, and the other to translate the destination address into the format required by the operator for routing purposes on their network.

The presence of these files is essential to ensure that the message parameters will be transformed in a way that is understood by the SMSC and the network, which prevents the message from failing because of a wrong translation.

4.5.8 Lawful Interception

The lawful interception plays an important role in detecting suspicious activities. It is used by the authorities or some official parties and is enabled by the operator to allow the interception of short messages sent/received by certain individuals.

To enable it, an ESME should be available to receive the forwarded message. That is why some ESME connections are critical and should be highly monitored.

4.5.9 Tracking

All the transactions made on the SMSC (MO, MT delivery attempt, Alerts, ESME traffic...), are logged into a history file. Each transaction line contains all the parameters for a transaction. The presence of this file is very important for later investigations in case certain messages fail to be delivered, and it's the source of the creation of the CDR files used to perform the billing.

4.5.10 Auto-Signature

The subscribers can define a signature which will be automatically appended to all messages they send. The auto-signature is handled by an external application (smsc_signature).

In this thesis, the main focus is on these processes and features, being the most important ones. But also many other processes that were not mentioned in this chapter are included in the study. In the next chapter the implementation of the decision tree algorithm on the SMSC server will be discussed.

Chapter 5

The Implementation

In the first part of this experiment, the study was based on offline analysis done on prerecorded data. If the purpose is to detect anomalies in real-time, the algorithms used will have additional constraints and requirements, related to the performance of the algorithm and the amount of data on which the study should be done.

Experts in this field were interviewed, people who are working closely with servers, to better understand the strategies that are implemented, and what additional improvements might be of good use. Their feedback helped in putting together the concepts and cases that will be explained in this chapter.

The data was gathered from a real operating server, the SMSC, and it contains many attributes related to hardware, software running on the server, external connections. The values of these attributes were taken at specific time intervals.

One disadvantage of this would be that the algorithm will not be able to detect anomalies that are fast to happen and that occur between two consecutive selections of the data.

The anomalies that might be caught on the system can have different reasons, from software failures because of programming bugs, to hardware issues, or they can be related to high system load or even configuration errors. This study tries to cover as much as possible all the parameters that can be related to all types of failures or anomalies.

5.1 Server Related Studies

Before providing our own approach, it is important to check what other studies have proposed and implemented in this topic. This section shall present a summary of some of the approaches and methodologies that were done so far on server data monitoring and analysis [41][42][43][44][45][46].

One of the studies tried to forecast database disk space requirements by using linear regression analysis. In this study, regression points to the fact that some variations in disk space usage were observed on the short term, but this growth on the long term was linear-looking and therefore easily predictable.

Another study focuses on combining monitoring of the service level with the normal monitoring of the resource usage. The purpose is to find the best thresholds that divide two bivariate time series in a way to maximize the mutual information between them.

A practical study considered data that was collected by network monitoring agents and applied data mining techniques to it. The goal was to find the causes that affect the performance. The concepts were applied to real-world data of a tracing application for aircraft. Values for 250 parameters related to system hardware values were collected at regular intervals, for a period of 2 months. Decision tree algorithm, top n algorithm, rule induction algorithm, and inductive logic programming were used, and many unexpected problems were detected.

Also, dynamic syslog mining was proposed to detect failure symptoms in computer devices. The steps consist of using a mixture of Hidden Markov Models to represent the syslog behavior, to adaptively learn the model using an online learning algorithm in combination with dynamic selection of the optimal number of mixture components, and to give anomaly scores using universal test statistics with a dynamically optimized threshold.

The “universal test statistics” uses the combination of Shannon information and event compression efficiency to give an anomaly score.

Many companies implemented anomaly detection in their systems. Among them the Intel IT’s Business Intelligence platform whose infrastructure goes through a three-stage process: It collects and analyzes log files from proxy servers, domain name servers (DNS), Dynamic Host Configuration Protocol (DHCP), Active Directory databases, and other systems. It extracts data from server logs, security sensors, intrusion systems, and management platforms, and parses these contextual data at more than a million events per second. And it then generates reports and workflow automation using custom algorithms that can identify unusual events.

Having considered the previous studies and other previous work, it is realized that there are some important issues to be tackled. This will be done in the next section.

5.2 Important Issues to Address

In this context, there are some important issues to tackle in our study:

- The number of attributes: Since an object may have many attributes, it may have anomalous values for some attributes but it can also be anomalous even if none of its attribute values are individually anomalous.
- The perspective (global versus local): The object can look anomalous with respect to all other objects, but not with respect to its local neighbors.
- The degree of anomaly: Some objects are more anomalous than others, in the sense that their anomaly can be very critical to the health of the system. That is why it is preferable to assess the degree of anomaly of a certain object (by using an anomaly score or any other similar method).

- “One at a time” versus “Many at once”: decision should be made concerning whether it is better to remove the anomalous objects one at a time or identify a collection of anomalous objects together?
- Evaluation: A good measure should be found to evaluate the process of anomaly detection (precision, recall, FP-rate, accuracy...).

The first challenge was to understand the system and everything related to it. An extensive study was conducted, many documents and technical descriptions were read in order to grasp the most important concepts and functionalities of the SMSC.

The second challenge was to decide which data to gather and the way to do it. The system can generate thousands of different information, but not all of this information is useful or essential for anomaly detection.

"The challenge with search is that you fundamentally need to know what you're searching on. Given the explosion of data, it's humanly impossible to know everything about your data" says Sanjay Sarathy, chief marketing officer (CMO) of machine data analytics specialist Sumo Logic [5].

This is why at first too many variables were gathered, but with the progress of the study, many of them were eliminated for more simplicity and accuracy.

Another challenge came up after gathering the data, concerning which machine learning algorithm to choose to tackle this problem. The first objective was to balance the accuracy of the prediction and the speed at which the entire data could be trained on and classified. In our case, the higher importance was given to the accuracy, since it is necessary to have at each step a correct classification about the status of the system, whether it is normal or anomalous, because this decision will have an impact on the traffic and the functioning of the server. In the following sections the main steps that were conducted will be summarized.

5.3 Initial Data Selection

As mentioned previously, the purpose of this study was to gather from the SMSC server the maximum number of “useful” parameters that affect, in a way or another, the status of the server and the performance in general.

The first step was to collect all the possible data, and to find a fast way to do that. Around 500 attributes were collected.

In the following, some of the parameters that were selected and that cover almost everything on the system are listed:

Central Processing Unit (CPU) statistics and input/output statistics for devices, partitions and network filesystems (NFS): These values help in monitoring the system input/output device loading by observing the time the devices are active in relation to their average transfer rates. Among these: the percentage of CPU utilization that occurred while executing at the user level (application), the percentage of CPU utilization that occurred while executing at the system level (kernel), the number of transfers per second that were issued to the device (i.e. the number of I/O requests to the device), the amount of data read from the device expressed in number of blocks per second (blocks are equivalent to sectors with kernels 2.4 and later and therefore have a size of 512 bytes), the amount of data written to the device expressed in number of blocks per second, the total number of blocks read, the total number of blocks written, the number of read requests that were issued to the device per second, the number of write requests that were issued to the device per second, the number of blocks read from the server by the NFS client via an NFS READ request, the number of blocks written to the server by the NFS client via an NFS WRITE request, and many others.

Memory usage (the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel): the amount of virtual memory used,

the amount of idle memory, the amount of memory used as buffers, the amount of memory used as cache, the amount of inactive memory, the amount of active memory, the amount of memory swapped in from disk (/s), the amount of memory swapped to disk (/s), the number of blocks received from a block device (blocks/s), the number of blocks sent to a block device (blocks/s), the number of interrupts per second, including the clock, the number of context switches per second, and others.

The percentage of the used/available disk space in each partition of the hard disk.

The connection with the network hosts, namely the other SMSCs in the same cluster, by sending multiple ICMP ECHO_REQUESTs to these hosts which respond by an ICMP ECHO_RESPONSE. Here also these requests and responses were transformed into values for the corresponding parameters.

The applications: the processes running on the servers, the number of processes waiting for run time, the number of processes in uninterruptible sleep, and many others. The most important processes were introduced in the SMSC chapter. They include for instance: the SMSC main process, the distribution list process, the MNP process, the AIM process (which handles connections with the ESMEs), all the synchronization processes, the REU handler process (which handles the connection with the REU), the apache process (which enables the use of the web interface), the heartbeat, and other processes. Parameters were created to hold values related to whether these processes are running or not, and the number of instances of each process. Also, the number of defunct processes and their nature were included.

The cores present on the server: A process dumps core when it is terminated by the operating system due to a fault in the program. This mainly occurs when the program accesses an invalid pointer value.

The DAT and CDR files: Whether the DAT file is writing correctly, and the CDR file is being created as it should.

The connections with all the external entities.

General information about the mysql database: its size, the connection to it, its current state (running or not), the number of messages saved in the database, the normal and the broadcast messages, the number of messages waiting to be delivered in a later stage (The sending of these messages will be retried after a certain period of time, since in the first time it failed for some valid reasons), and here will be counting the records in three important tables: messages, waiting, and future.

The queues: The number of messages stuck in the different queues. All processes write/read to/from queues. It is very essential to keep track of the state of the queues in order not to slow down the execution.

The counters: The value of the different important counters, which count the number of received messages, failed messages, those that were successfully sent.

5.4 The Script Used for Data Selection

The operating system running on the SMSC is Linux (Redhat). In order to collect the data mentioned above (and other related data), a shell script was written and tested. This script included around 500 parameters (attributes) as mentioned previously, and it was set to run automatically every 1 minute and save the gathered data in a csv file. A file per day is produced, and the timestamp is added next to each entry containing all the values for the attributes. Some parts of this script can be found in Appendix A.

The result file was checked after 3 days and it was decided to run the script every five minutes, since it contained many duplicate records due to the stability of the system at that time and the fact that most of the parameters don't vary a lot in one minute. The script was kept on the server for 30 days. After that period, the CSV file that included more than 8600 records was taken in order to analyze it and apply to it the chosen data mining algorithm.

5.5 Cleaning the Data or Data Preprocessing

After having the file ready, the data had to be cleaned. Some records were duplicate (although it was chosen to run the script every five minutes), others had missing values. So it was essential to decide whether to delete the corresponding record or to find a logical way to fill the missing parameters with data.

Next, the main steps required for data cleaning are described.

5.5.1 Handling Duplicates

Figure 5-1 below shows the pipeline that handles duplicates in the data set. The original data set was processed, all duplicated were removed, and a weighting factor was assigned to each record in the new data set, which is equal to the number of identical records in the original data set. The algorithms operate on the new data set and the scores are mapped to produce the result set.

This procedure has a main advantage of significantly reducing the number of records, which has a positive influence on the performance.

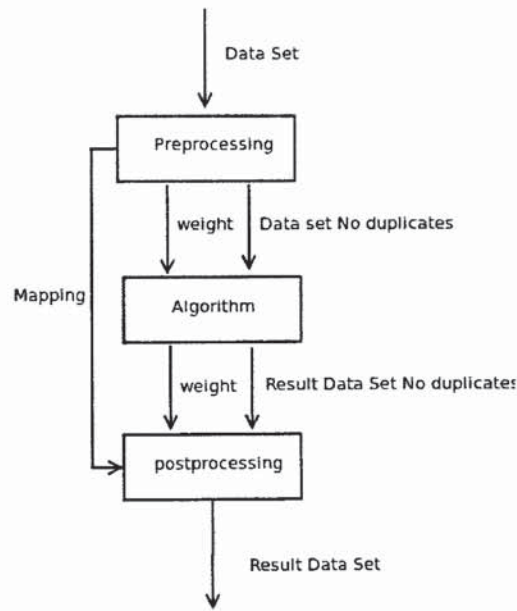


Figure 5-1 Handling Duplicates

5.5.2 Handling Missing Values

For the records that contain missing values for some attributes, another algorithm was used to fill these values logically. The previous 10 records were taken into consideration, and a formula was applied to find the pattern that relates them. If the values are constant, the same number is added. If they are being incremented constantly, the same increase is followed...

For records that contain more than 3 missing values (which are very few), the whole record is deleted.

5.6 Adding Anomalous Records

Most of the records in the result file correspond to normal cases, which is logical since the frequency of anomaly occurrence is very minor with respect to the normal cases. In

order to be able to use a classification algorithm, the file needs to contain a balanced number of normal and anomalous cases.

For this reason, another script was used to generate similar data (more records) with some various changes based on historical behavior of the system and issues that were previously encountered, on the records present in the file, and the possible values that the attributes can have.

The final number of records was limited to 6000, which will go through the labeling process.

5.7 Differences in Attribute Types

The attributes (indicators) that are gathered in the study differ from many points of view. Some of them behave in a consistent way, in a sense that their values remain almost the same or in a certain small range. Others can be somehow unstable, and perform some irregular changes at some point. Some can remain always constant but exhibit a huge change of value at specific times.

Some indicators are a measurement in percentage, so their value will certainly be between 0 and 100. Others can also have a limited range, for example the amount of free disk space, since it has to be between 0 and the maximum size of the disk. On the other hand, some indicators can have no boundaries for their possible values, and these would be harder to analyze.

The way the indicator's value changes affects whether the situation is normal or anomalous. For example, if the value of a certain indicator increases rapidly, this might not be an alarm of something anomalous. Whereas if its value decreases suddenly, it might be considered as a danger to the system. So the variation of an indicator's value is not always considered as a sign of anomaly presence, but the direction of this change can be. [40]

5.8 Correlations Between Attributes

During the study, one of the observations was that some indicators have correlations between them, in the sense that their values are somehow related, and they change in the same direction or following a certain pattern.

Figure 5-2 shows the relation between the number of instances of the SMSC process and the SMSC alive counter (that keeps incrementing as long as the process is running and executing). As it can be seen, the value of the alive counter is directly proportional to the number of instances of the SMSC process. For this reason, one of these two attributes can be deleted. The same logic applies to few other attributes.

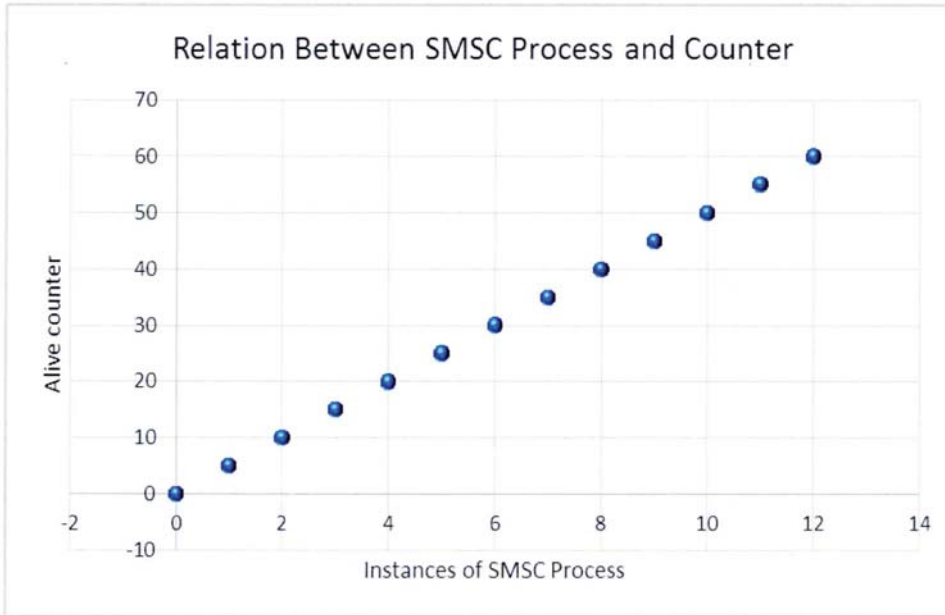


Figure 5-2 Relation Between SMSC Process and Counter

Figure 5-3 shows also an X-Y scatter graph representing the relation between the number of messages in the “Messages” table in the database, and the MySQL size. It can be seen that the relation between the two is almost linear, i.e. the MySQL sizes increases with the number of messages stored in the “Messages” table (of course with other factors also since they are not perfectly proportional). The same concept was tested with different other attributes in order to reduce their number and remove unnecessary indicators that do not give additive information.

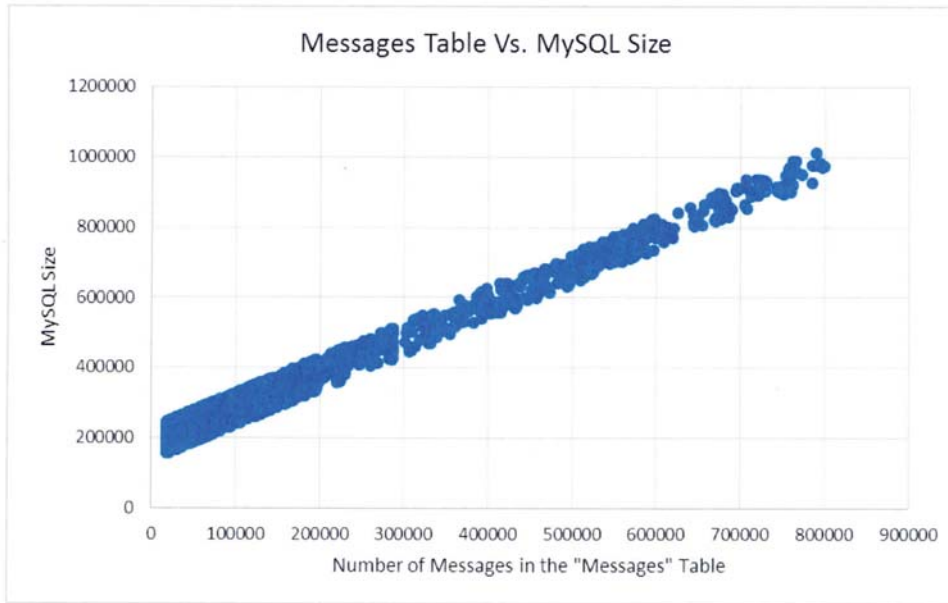


Figure 5-3 Relation Between Messages Table and MySQL Size

5.9 Seasonality

The attributes can have some logical patterns based on a certain periodicity. This seasonality is caused by the fact that the indicator values are sometimes influenced by the hour of the day, the day of the week, special holidays or event-based days. For example, having a high CPU usage during an activity on the system is not an anomaly, but having the same value at another time can be considered as anomalous. Also, other attributes can have a high value on weekends or on high broadcast traffic hours, which is considered normal, but not on periods with low traffic.

The concept of seasonality was studied for the three-month period of the data gathering. Figure 5-4 shows a sample of the average CPU usage for 6 days of the first week, where the average CPU usage per hour was calculated. It can clearly be seen that the CPU usage has higher values between 11 am and 1 pm on Wednesday, Thursday and Friday (with a high value for 2 pm on Friday), and even higher values between 8 am and 2 pm on Saturday. Since this pattern keeps repeating every week, it will not be considered as a sign of anomaly, and it is due to a known increase in the traffic during these periods, along with huge broadcast from different content providers.

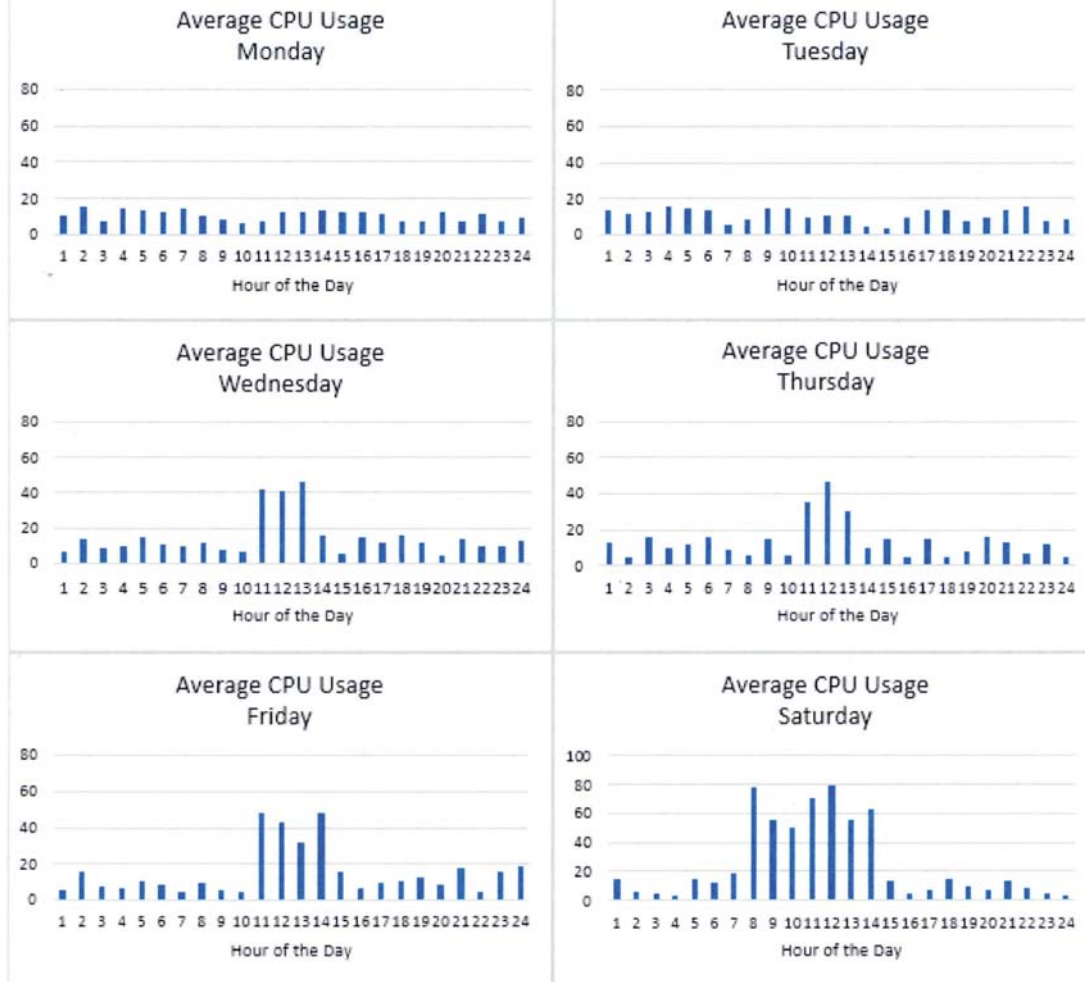


Figure 5-4 Average CPU Usage For 6 Days

Some of the anomalies can be easily detected by using constant thresholds, but others cannot be found this way. Now if their behavior is the same every week or month for example, they can be detected by using time-dependent thresholds, but sometimes this seasonal behavior concept cannot be applied to the indicators.

After going through all these correlations, the number of attributes was lowered to 56, by removing those that are of minor importance, that are considered somehow duplicates in the information they provide, or that don't highly affect the state of the system.

5.10 Data Labeling

This part is considered one of the hardest, since it requires a lot of time and focus to be able to understand each case and label it manually.

Commonly, selected records are divided between two classes, normal and anomalous. Here it was chosen to divide them into three classes: normal, warning and critical (anomalous), where the warning class includes records that are considered a middle state (not totally normal, and not anomalous yet). This class is used to warn that the system is probably going towards an anomalous state so that the appropriate action will be taken ahead of time, since most of the cases pass through this warning state before being anomalous.

The labeling process had to take into consideration the type of anomaly (point, contextual, collective). To enforce the contextual anomaly concept, the data set was divided into subsets taking the time as a contextual attribute, after analyzing the different types of "periods" that the system goes through (the concept of seasonality). All these concepts were gathered, notes were taken, records were divided into subsets, and decisions were made to classify each instance as "normal", "warning" or "critical".

Now having the labeled file, the purpose is to work with it, apply to it the decision tree algorithm, and perform some tuning and modifications in order to improve the accuracy. For this to be done, a software is needed. In the following section the WEKA software which was used in this thesis will be presented.

5.11 WEKA Software

WEKA is a machine learning software created by researchers at the university of Waikato in New Zealand [48]. It is a Java based open source collection of many data mining and machine learning algorithms and it provides facilities for all the steps involved in solving these problems. These facilities include data conversion, pre-processing, classification, clustering, association, categorization, and it also provides a GUI on supporting systems, which makes it very easy in terms of understanding the flow of data, and visualizing the results. WEKA contains 49 data preprocessing tools, 76 classification/regression algorithms, 8 clustering algorithms and many others.

On the other hand, it has a simple Command Line Interface (CLI) which is lightweight in terms of memory, and which provides much more scope for dealing with very large files.

Weka was one of the most cited, used and recommended machine learning tool in the studies performed previously [30]. It is very well documented and it is very convenient to understand and customize.

Figure 5-5 shows the first GUI that the software displays when it runs. In this thesis, the main interest is in the “Explorer” section which is concerned with preprocessing, attribute selection, learning and visualization. Data files can be loaded to WEKA, having formats such as ARFF, CSV, C4.5 and binary, and the user can also import data from URLs or SQL database.

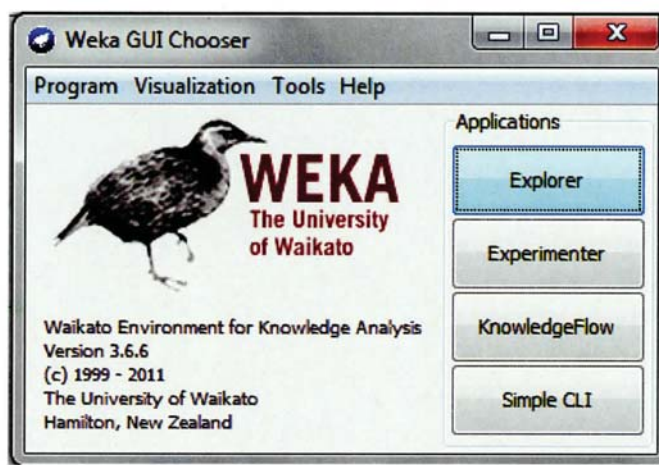


Figure 5-5 WEKA GUI

5.12 Data Manipulation

In this research, many experiments were conducted using different cross-validation levels, percentage split values and pruning strategies. The first purpose was to decide which strategy should be used in order to have a smaller but meaningful tree, and a higher accuracy.

5.12.1 Choosing the Typical Number of Records

In order to understand the impact of the number of records on the results, different tests were considered using different file sizes. These cases were treated using the 10-fold cross validation. In the following, the impact of the number of records on three different result parameters will be studied: the training time, the tree size and the accuracy.

Figure 5-6 shows the time taken to train the dataset in relation with the number of records in the dataset. It can be seen that for very small file sizes, the training time is negligible. And whenever the size increases, the training time also increases. For a file containing 6000 records, less than 1.2 seconds are needed to output the results.

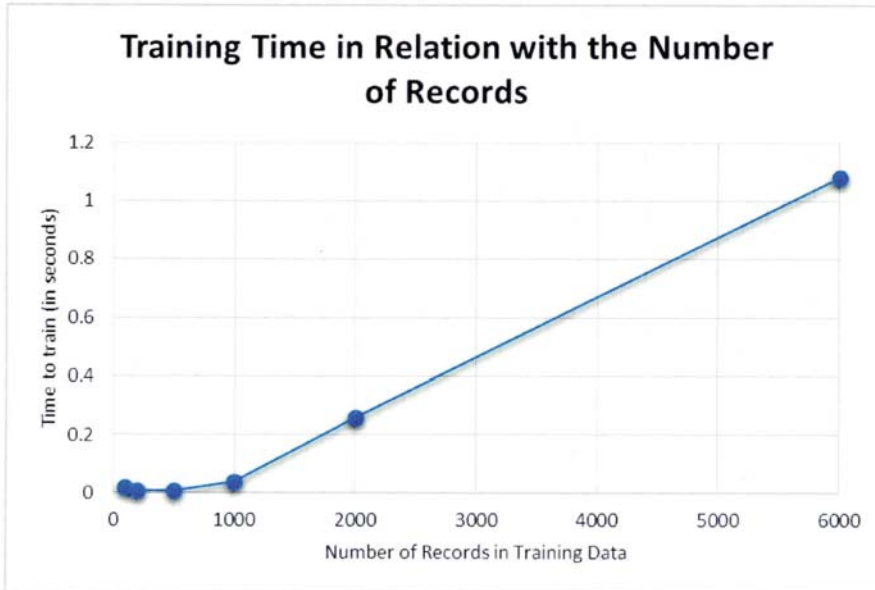


Figure 5-6 Training Time Vs. Number of Records

The increase in the training time is almost exponential and it is related to the number of features in the training set. The algorithm goes through all the instances for each feature in order to calculate the entropy. Therefore, the training time is largely proportional to the number of features in the training data and the number of instances.

5.12.1.2 The Tree Size

Figure 5-7 shows the effect of the number of records on the size of the tree and the number of leaves. Typically, as the number of records increases, the size of the tree increases and so does the number of leaves. But taking a very small tree would negatively affect the classification, and would not cover all the attributes that should be included in the decision making.

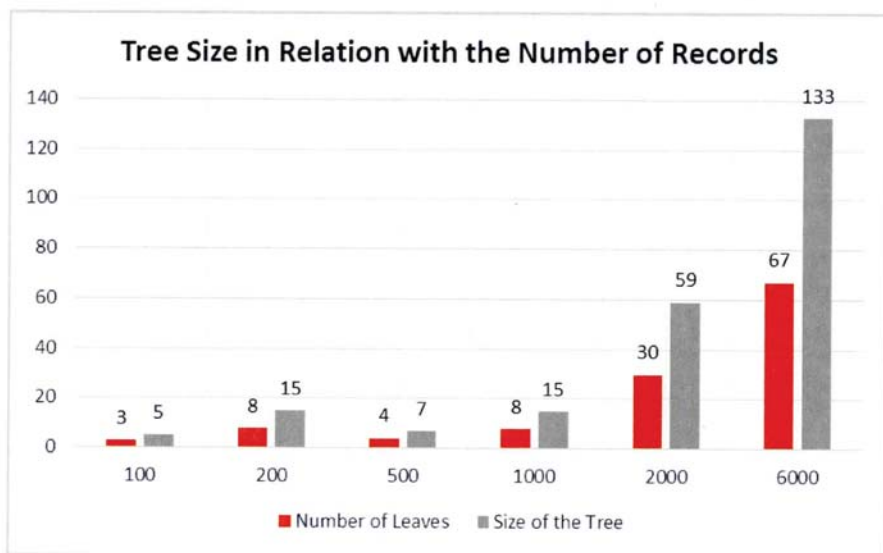


Figure 5-7 Tree Size Vs. Number of Records

5.12.1.3 The Accuracy

The same concept applies to the accuracy. Figure 5-8 shows the relation between the number of records and the accuracy. Small files don't lead to a trusted precision, and it was

seen that a 6000 records file gives an accuracy close to 99% whenever applying to it the decision tree algorithm explained before.

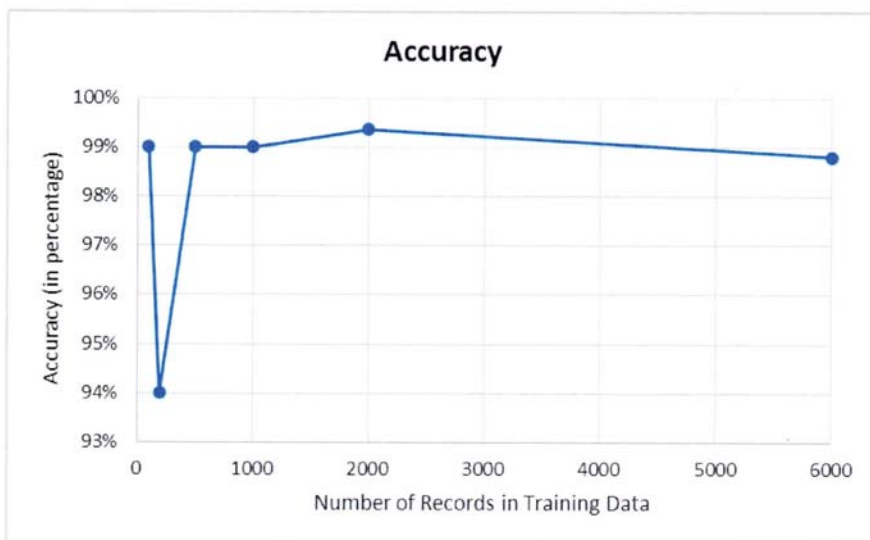


Figure 5-8 Accuracy Vs. Number of Records

5.12.2 Between Cross-Validation and Percentage Split

The number of folds of cross validation was changed, and it was shown that it also has an effect on the training time of the tree: The higher the number of cross validation folds, the more the training time. The files were tested for 10, 15 and 20 cross-validation. This was also tested for different types of percentage split. 10-fold is the most commonly used, and after these tests it was seen that it is also the best to be used in this thesis.

The previously mentioned cases are summarized in Table 5-1, along with a comparison of the same cases but using a percentage split of 70%. The aim is of course to have a high prediction accuracy, but also to test the effect of preprocessing and pruning techniques on the training time, to have an acceptable training time, and to have a tree that somehow covers all scenarios, i.e. includes most of the attributes.

Table 5-1 Cases with Different Training Data File Size

Using Cross Validation = 10					
Test	Number of Records in Training Data	Time to Train (s)	Number of Leaves	Size of the Tree	Accuracy
1	100	0.02	3	5	99%
2	200	0.01	8	15	94%
3	500	0.01	4	7	99%
4	1000	0.04	8	15	99%
5	2000	0.26	30	59	99.37%
6	6000	1.08	67	133	98.82%

Using Percentage Split = 70%					
Test	Number of Records in Training Data	Time to Train (s)	Number of Leaves	Size of the Tree	Accuracy
7	100	0	3	5	100%
8	200	0.01	8	15	91.67%
9	500	0.01	4	7	99%
10	1000	0.03	8	15	99%
11	2000	0.23	30	59	99.22%
12	6000	0.96	67	133	98.83%

It can be seen from the table that all cases have an accuracy higher than 98.8%, which is very acceptable as a first step. For small files used for training, the accuracy is nearly 100%, but the tree is small, containing 3 leaves, which limits the number of attributes to a maximum of three, thus completely ignoring the other attributes. This means that although the accuracy is very high, but it will have a lot of false positives and will not be able to catch all anomalies since the criteria is only based on very few attributes.

The bigger the dataset, the bigger the tree, but the more precise it is, having a very acceptable training time compared to small datasets, and keeping a high value for accuracy.

As stated before, there is no common best solution for all decision tree problems. The amount of cross validation, pre-processing, feature selection, filtering, etc. depends entirely on the kind of data being handled.

It was decided to use 6000 records as training data (of course having a larger set will yield more precise results), and the 10-fold cross validation was used since no big difference was noticed when compared to the 70% split.

Now more tuning should be done on the data in order to increase the accuracy and improve the results.

5.12.3 Applying Tuning Concepts

Now having decided to use the file containing 6000 records along with 10-fold cross validation, it was time to manipulate the data in order to have better results and higher accuracy.

The main objective was trying to work with the attributes and perform some tuning in order to improve the results.

5.12.3.1 Combining Multiple Attributes into One

5.12.3.1.1 Disk Space Attributes

First, the disk space values were combined into one. The original script gives the disk space usage in each partition of the disk, in this case five partitions. These five attributes were replaced by one, which contains in itself information about all the partitions. They can

be replaced by an attribute showing their average (this way any sudden increase in the disk space of any partition will be noticeable, since the average would also increase). Or they can for example be replaced by the maximum value among the disk space of the five partitions. In this case, it can be sure that no partition will become nearly full without taking that into consideration in the analysis, but this doesn't give any information about whether another partition having a lower used disk space is not encountering an increase in its value (which also might indicate an anomaly).

Making this modification and running the algorithm gave a tree with 57 leaves (10 less than the original one), and with an accuracy of 98.93% (0.11% more).

5.12.3.1.2 Ping Attributes

Trying to combine all variables related to pinging the other nodes in the cluster didn't lead to better results. A tree with a bigger size (123) was obtained, but with less leaves than the original one, and with the same original accuracy value (98.82%).

5.12.3.1.3 Other Attributes Combination

Many other attribute manipulations were done such as: combining both disk space usage and ping usage, removing attributes that are somehow correlated to other attributes (like mysql size which is related to the number of messages in the waiting table and the messages table...) or attributes that were found to have no effect on the performance or the result (CDR, brdcast and normal messages...).

5.12.3.2 Replacing Attribute Values by One of Two Values

For some variables, the value (no matter what it is) can be replaced by only one of two values. For example, it might not be important to know how many instances of a process are running, but rather whether this process is running or not. Same logic applies to the

defunct processes found on the system and other attributes. Also, considering the running processes, this method does not have to be applied to all of them. For some it might be useful to know if with time the number is decreasing, for others it might be important to have at least one instance running. For the latter case, a script was run to replace by 1 every value greater than 1 (0 means not running, and 1 running).

This concept gave an improvement, with a tree formed of 56 leaves, and of size 111, with an accuracy of 99.12%. And applying it to some counters also improved the accuracy to 99.22% and gave a smaller tree.

The results of some of the tests performed (mostly mentioned in the sections above) are summarized in Table 5-2.

Table 5-2 Modification Cases

	CASE	Accuracy (%)	LEAVES	SIZE OF TREE	TIME TO BUILD (Sec)
1	No Modification	98.8333	67	133	0.56
2	Disk Space Mod.	98.9333	57	113	0.46
3	Ping Mod.	98.8167	62	123	0.49
4	Disk Space and Ping Mod.	98.9	62	123	0.33
5	Removing 1 Attribute	98.9833	61	121	0.34
6	Removing 2 Attributes	99.1167	56	111	0.3
7	Changing variable (processes)	99.1167	56	111	0.3
8	Changing variable (defunct)	99.1333	62	123	0.29
9	Changing counters' values	99.2167	52	103	0.41

To note that the cases in the above table are additive, i.e. for example case 6 is a case where modifications to the disk space variables and ping variables were performed, and where three attributes were removed.

5.12.3.3 Applying the Concept of Ranges

The next step was to try to tune the values of some attributes, especially those that have a lower and an upper bound such as the CPU usage, the memory usage, and other variables that are expressed in terms of percentage (value between 0 and 100). The idea here is that having few values for a certain attribute (4 or 5) might lead to a smaller tree and a higher accuracy than having 100 possible values for this attribute.

Many cases were tackled, that include many attributes, and that differ by the ranges taken into consideration. For example, in the first case, the possible range was divided into five parts:

- If the value is between 0 and 20 it will be assigned a value of “1”
- If it is between 20 and 40 it will be given a value of “2”
- If it is between 40 and 60 it will be given a value of “3”
- If it is between 60 and 80 it will be given a value of “4”
- If it is between 80 and 100 it will be given a value of “5”

Here the focus will not be on the time taken to build the model since it is always within parts of a second, so a very acceptable value.

First, the value of CPU load was transformed into one of the five possible values mentioned above.

Then the same concept was applied to both CPU and IO, then other attributes were added to them... not all cases of tuning gave better results, but rather some of them caused a decrease in the accuracy instead of improving the learning.

Also, many trials were done to change the number of ranges used and their boundaries, based on specific values of some attributes that might lead to an anomaly.

For example, 0-25-50-75-100 means that the variable is divided into four ranges, where the first range (between 0 and 25) is given a value of 1, the second range (between 25 and 50) is given a value of 2, and so on.

In the following Table 5-3, some of the cases that were tackled will be presented, along with the results obtained when applying each one of them:

Table 5-3 Tuning Cases

	CASE	Accuracy (%)	LEAVES	SIZE OF TREE
1	1 Att - 0-20-40-60-80-100	98.8167	62	123
2	2 Att - 0-20-40-60-80-100	98.3167	72	143
3	3 Att - 0-20-40-60-80-100	95.2333	141	281
4	0-40-60-80-90-100	99.1333	69	137
5	0-50-70-80-90-100	99.05	76	151
6	0-60-80-90-95-100	99.05	76	151
7	0-25-50-75-100	95.1167	153	305
8	0-25-50-75-100	92.9167	222	443
9	0-40-60-80-100	94.1333	153	305
10	0-60-80-90-100	97.6667	108	215
11	0-80-90-95-100	97.3667	93	185
12	0-30-60-100	91.3333	199	397
13	0-60-80-100	93.2667	159	317
14	0-80-90-100	98.3333	89	177
15	0-80-100	92.6333	161	321
16	0-90-100	94.15	98	195
17	DS only - 0-80-90-100	99.3833	51	101

The best results were observed in case 17, this is why it was adopted.

Now some of the statistics obtained by WEKA after training the result file will be presented.

5.13 Statistics

In this section, the statistics obtained will be shown.

5.13.1 Three-Class Model

In the following (Figures 5-9 and 5-10), some statistics of the model obtained when using the final result file are presented (after applying to it all the modifications explained previously).

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      5963      99.3833 %
Incorrectly Classified Instances    37        0.6167 %
Kappa statistic                    0.9907
Mean absolute error                0.0057
Root mean squared error            0.0632
Relative absolute error            1.2775 %
Root relative squared error        13.4493 %
Total Number of Instances         6000
```

Figure 5-9 Statistics - Case 1

It can be seen that the classes' accuracies have high values. The TP (True Positive) rate for the normal class is 99.9%, for the critical class 99.4% and for the warning class 98.8%. All F-measures and precisions are above 99%.

```
=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      0.999   0.002   0.996     0.999   0.997     0.999   Normal
      0.988   0.004   0.991     0.988   0.99     0.996   Warning
      0.994   0.003   0.994     0.994   0.994     0.997   Critical
Weighted Avg. 0.994   0.003   0.994     0.994   0.994     0.997
```

Figure 5-10 Detailed Accuracy - Case 1

Figure 5-11 shows the three-class confusion matrix.

```
=== Confusion Matrix ===
      a    b    c  <-- classified as
2136   3    0 |  a = Normal
  8 1734   13 |  b = Warning
  1   12 2093 |  c = Critical
```

Figure 5-11 Confusion Matrix - Case 1

Three normal instances are misclassified as warning, 13 warning instances classified as critical, 12 critical instances classified as warning. All these, although misclassified, don't have a negative effect on the result. Also, the 8 warning instances classified as normal can be considered as non-affecting the result negatively, since the warning class is only used to indicate a possibility of anomaly. So even if it's classified as normal, it won't be a problem. Therefore, effectively, only 1 instance here out of 6000 is seriously misclassified (classified as normal while it's critical). This leads to an "effective" accuracy of 99.98%.

Of course having a larger training data set will lead to even better results.

5.13.2 Two-Class Model

In all previous studies, they would use only two classes (normal and anomalous). Therefore, another experiment was carried out in order to see whether having only two classes would lead to better results. The same final file used for the three-class case was taken, replacing the "warning" and "critical" classes by "anomalous". So now two classes exist: normal and anomalous.

Running the decision tree algorithm lead to a very smaller tree (with 27 leaves instead of 51, and a size of 53 instead of 101), with a better accuracy of 99.77%. The statistics are shown in Figure 5-12.

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      5986           99.7667 %
Incorrectly Classified Instances     14             0.2333 %
Kappa statistic                     0.9949
Mean absolute error                 0.0029
Root mean squared error             0.0462
Relative absolute error             0.6278 %
Root relative squared error         9.6402 %
Total Number of Instances          6000

```

Figure 5-12 Statistics - Case 2

All the accuracy values are above 99.7% for both classes, as shown in Figure 5-13.

```

=== Detailed Accuracy By Class ===

      TP Rate  FF Rate  Precision  Recall  F-Measure  ROC Area  Class
      0.998    0.033    0.995    0.995    0.997    0.999    Normal
      0.997    0.002    0.999    0.997    0.998    0.999    Anomalous
Weighted Avg.  0.998    0.002    0.998    0.998    0.998    0.999

```

Figure 5-13 Detailed Accuracy - Case 2

Figure 5-11 shows the two-class confusion matrix.

```

=== Confusion Matrix ===

      a    b  <-- classified as
2135    4  |    a = Normal
   10 3851 |    b = Anomalous

```

Figure 5-14 Confusion Matrix - Case 2

Four normal instances are classified as anomalous (which has no negative effect on the performance), while 10 anomalous instances are being considered as normal. Although the two-class decision tree gave a higher accuracy, but it was decided to adopt the three-class since the effective accuracy (calculated based on our preferences relative to the problem here) was higher than that of the two-class (99.98% versus 99.83% here).

5.14 Transforming the Model into a Script

After deciding which model to adopt, the decision tree was transformed into rules, and these rules were transformed into a script.

WEKA gives the ability to do this transformation and to output a source code. Some parts of this code are shown in Appendix B. The code at hand was written as a shell script in order to be understood by the linux operating system. The script was set to always run on the SMSC server.

5.15 The Alarm System

An alarm system was created. The original script that gathers the data would run every minute. The gathered data acts as an input to the script that describes the model. After being executed (which takes few milliseconds), the script classifies the record as normal, warning, or critical.

If it is normal, no action will be taken. Otherwise, an alarm will be sent to a monitoring screen stating whether it's a warning or it's a critical state. Also, along with the pop-up window that shows the state of the system, an alarm sound will be played.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

All systems and servers generate a huge amount of data that can be used to give information about the state of the system. Many studies were conducted to understand the problem of anomaly detection. Different techniques were implemented, and the advantages and disadvantages of each were explored.

The purpose is to catch any error or anomaly before it causes harm to the system. A practical use case would be to decide at which point some system components might need upgrade, for instance adding more memory or disk space. If these were to be added early, useless additional costs would be created. And failing to add them at the appropriate time might result in loss of performance of the server or the inability to catch the errors.

Also, whenever a certain upgrade is performed, the system should be able to give a proof of the improvement of the performance. Another use case is to be able to estimate the load that an application can have on the system, before installing it, to see whether the server is able to cope with it or not.

All these and many others were the reason behind this thesis and the studies done in the anomaly detection field.

6.2 Main Contribution of this Thesis

In this thesis, an anomaly detection algorithm was implemented on a server that didn't apply any anomaly detection technique before. The decision tree algorithm was presented, and all the features of the SMSC server. Then the implementation was proposed,

the tuning, and the different tests that were performed to obtain the best results were explained.

Unlike older studies, the use of three-class decision tree was suggested, which has a higher effective accuracy than the two-class, and which is more practical since it sends an alert before the actual anomaly occurs. The model had a very high accuracy, and proved to be working as expected.

6.3 Possible Extensions and Future Work

The concept of anomalies in a distributed architecture was explained in an earlier chapter, but was not tackled in our study since the focus was on a single server. What could be done more is to extend the study to cover more than one server, to find ways to cover the anomalies mentioned in this context, and to detect any differences between two similar nodes in a cluster.

The problem with the thresholds is that the user has to define them. This works in an environment that has stable and infrequent deployments, like our case here (which relied somehow on a threshold concept during the labeling process). But when the environment is dynamic this causes many difficulties: The user has to continuously assess the suitability of those thresholds, and redefine them as needed, the normal conditions can change depending on specific time/day, and thresholds don't give a precise result when encountering an abnormal trend or behavior. Anomalous behavior will give an alarm if it occurs within threshold boundaries. But in most cases, by the time the threshold is reached, it will be too late. It's not a good practice to wait until an anomaly to occur to fix it. The system might not be broken yet, but it can be in the process of breaking. [47]

Therefore, some extensions of this study would be to improve the use of the concept of seasonality, and avoid being dependent on thresholds. The best way would be to apply

unsupervised techniques that continuously learn (even though the system in our case is somehow stable), and try to obtain higher accuracies than the results in this thesis.

Some other issues that should be considered in the future when working with anomaly detection:

- **Labelling of data.** Usually, the researchers assume that the recorded data is normal and can be directly used to build the normality model. But this isn't true, since these data can contain anomalous records. The data is not labeled, so if unsupervised techniques were to be used, a technique that deals with the data without assuming that it's 100% normal should be implemented.
- **Changes in the system.** The frequency of the updates' installation can influence the normality model. The algorithm used should be able to take into account the changes made to the system.
- **Architectural issues.** The collection of the data from the server influences itself the server performance, since this collection is consuming memory, CPU load and sometimes disk space... This fact should also be taken into consideration.
- **Algorithms and experts.** An instance can be classified as anomalous, but it can be a consequence of an anomaly. The algorithm does not always give an ultimate decision about the severity of an anomaly. That is why the presence of experts is always needed to give the final word about a certain situation.

Appendix A

Script to Gather the Data

```
#!/bin/bash

day=`date +%Y%m%d`

# counters part 1
...

# Percentage of CPU usage
CPU=`top -d 0.1 -n 2 | grep '^Cpu(s):' | tail -1 | awk '{print $5}'|sed 's/%.*/'`
CPU1=`echo $CPU | cut -d"." -f1`
LOAD=$((100 - $CPU1))

# iostat
DEV=sda
IOSTAT=/usr/bin/iostat

    #RedHat release 3
    IO_ACT=`$IOSTAT -d -x | grep -w $DEV | awk '{print $14}'`
    if [ -z "$IO_ACT" ]
    then
        #RedHat release 5
        IO_ACT=`$IOSTAT -d -x | grep -w $DEV | awk '{print $12}'`
    fi
    # %util is given with 2 digits after the floating point
    IO_ACT_NUM=`echo $IO_ACT | awk -F '.' '{print $1}'`

# memory usage - buffers
FREE_MEM=`/usr/bin/free | grep buffers/ | awk '{print $4}'`
```

```
TOTAL_MEM=`free | grep Mem | awk '{print $2}'`
```

```
FREE_MEM_PER=`echo "scale=2; $FREE_MEM/$TOTAL_MEM" | bc | /bin/cut -d "." -f 2`
```

```
USED_MEM_PER=`expr 100 - $FREE_MEM_PER`
```

```
# swap usage
```

```
FREE_PATH=/usr/bin/free
```

```
#RedHat release 3 and 5
```

```
SWAP_LINE=`$FREE_PATH -m | grep Swap `
```

```
TOTAL_SWAP=`echo $SWAP_LINE | awk '{print $2}'`
```

```
USED_SWAP=`echo $SWAP_LINE | awk '{print $3}'`
```

```
# %util is given with 2 digits after the floating point
```

```
SWAP_USAGE=`echo "scale=2; $USED_SWAP/$TOTAL_SWAP" | bc `
```

```
SWAP_USAGE_NUM=`echo $SWAP_USAGE | awk '{print $1*100}'`
```

```
#ping usage
```

```
COUNT=2
```

```
count1=`ping -c $COUNT tsmc1 | grep 'received' | awk -F,' '{ print $2 }' | awk '{ print $1 }'`
```

```
count2=`ping -c $COUNT tsmc2 | grep 'received' | awk -F,' '{ print $2 }' | awk '{ print $1 }'`
```

```
count3=`ping -c $COUNT barmpu1 | grep 'received' | awk -F,' '{ print $2 }' | awk '{ print $1 }'`
```

```
count4=`ping -c $COUNT barmpu2 | grep 'received' | awk -F,' '{ print $2 }' | awk '{ print $1 }'`
```

```
# disk space usage
```

```
Temp1=`df -P / | grep -v Filesystem | awk '{print $5}'`
```

```
Used1=`basename $Temp1 %`
```

```
Temp2=`df -P /boot | grep -v Filesystem | awk '{print $5}'`
```

```
Used2=`basename $Temp2 %`
```

```
Temp3=`df -P /data | grep -v Filesystem | awk '{print $5}'`
```

```
Used3=`basename $Temp3 %`
```

```
Temp4=`df -P /home | grep -v Filesystem | awk '{print $5}'`
```

```
Used4=`basename $Temp4 %`
```

```
Temp5=`df -P /var | grep -v Filesystem | awk '{print $5}'`
```

```
Used5=`basename $Temp5 %`
```

```
# ps usage
```

```
ps_smsc=`ps -ef | grep smsc$ | grep -v grep | wc -l`
```

```
ps_smsc_dl=`ps -ef | grep smsc_dl | grep -v grep | wc -l`
```

```
ps_aim=`ps -ef | grep aim | grep -v grep | wc -l`
```

```
ps_httpd=`ps -ef | grep httpd | grep -v grep | wc -l`
```

```
ps_sync_db=`ps -ef | grep sync_db | grep -v grep | wc -l`
```

```
ps_reu_handler=`ps -ef | grep reu_handler | grep -v grep | wc -l`
```

```
ps_smsc_mnp=`ps -ef | grep mnp | grep -v grep | wc -l`
```

```
ps_jsync=`ps -ef | grep jsync | grep -v grep | wc -l`
```

```
ps_jfsync=`ps -ef | grep jfsync | grep -v grep | wc -l`
```

```
ps_heartbeat=`ps -ef | grep heartbeat | grep -v grep | wc -l`
```

```
# core monitoring
```

```
...
```

```
# dat monitoring
```

```
...
```

```
# cdr monitoring
```

```
hour=`date -d '-1 hours' "+%H"`
```

```
...
```

```
# defunct monitoring
```

```
stat=`ps -A -ostat,ppid,pid,comm | grep -e '^[Zz]' | grep -v grep` > /home /temp/geo/defunct-file.log
```

```
stat1=`cat /home /temp/geo/defunct-file.log | wc -l`
```

```
stat2=`cat /home /temp/geo/defunct-file.log | awk '{print $4}'`
```

```
# database messages monitoring
```

```
...
```

```
Mysql_DB=smsc_6_0
```

```

MESS_COUNT=`echo 'select count(*) from messages;' | /usr/bin/mysql $Mysql_DB -u
Mysql_USER -pMysql_PWD | grep -v count`

# database waiting monitoring
WAIT_COUNT=`echo 'select count(*) from waiting;' | /usr/bin/mysql $Mysql_DB -u $Mysql_USER
-pMysql_PWD | /bin/grep -v count`

# database future monitoring
#FUTURE_COUNT=`echo 'show table status like "future"\G;' | /usr/bin/mysql $Mysql_DB -u
Mysql_USER -pMysql_PWD | grep -i auto_increment | cut -d "." -f 2`
FUTURE_COUNT=`echo 'select count(*) from future;' | /usr/bin/mysql $Mysql_DB -u
Mysql_USER -pMysql_PWD | /bin/grep -v count`

# broadcast and normal messages
SMSCLISTMON=/home/bin/smsclistmon
BRD_MSG=`$SMSCLISTMON -q -t brdct | grep msg | cut -d "=" -f 2`
NORMAL_MSG=`$SMSCLISTMON -q | grep msg | cut -d "=" -f 2`

# MySQL disk mon
mysql_size=`du -s /var/mysql/innodb | awk '{print $1}'`

# MySQL conn mon
...
SQL_CURRENT_CONN=`echo "$SQL_CURRENT_CONN - 5" | bc`

SQL_CONN_RATIO=`echo "$SQL_CURRENT_CONN * 100 / $SQL_MAX_CONN" | bc`

# MySQL mon
MySQLstatus=`ls -l /var/lib/mysql/mysql.sock | wc -l`

# queues

```


...

counters part 2

...

```
dsmsc=`echo "$csmsc2 - $csmsc1" | bc`
```

```
daim=`echo "$caim2 - $caim1" | bc`
```

```
dmnp=`echo "$cmnp2 - $cmnp1" | bc`
```

```
dreu=`echo "$creu2 - $creu1" | bc`
```

```
dreua=`echo "$creua2 - $creua1" | bc`
```

```
dsyncdb=`echo "$csyncdb2 - $csyncdb1" | bc`
```

```
djsync=`echo "$csync2 - $csync1" | bc`
```

```
djsyncc=`echo "$csyncc2 - $csyncc1" | bc`
```

```
djagtx=`echo "$cjagtx2 - $cjagtx1" | bc`
```

```
D=`date +%Y%m%d%H%M`
```

```
echo
```

```
"$D,$LOAD,$IO_ACT_NUM,$USED_MEM_PER,$SWAP_USAGE_NUM,$count1,$count2,$count3,$count4,$Used1,$Used2,$Used3,$Used4,$Used5,$ps_smsc,$ps_smsc_dl,$ps_aim,$ps_httpd,$ps_sync_db,$ps_reu_handler,$ps_smsc_mnp,$ps_jsync,$ps_jfsync,$ps_heartbeat,$score_result,$dat1,$dat2,$cdr,$stat1,$MESS_COUNT,$WAIT_COUNT,$FUTURE_COUNT,$BRD_MSG,$NORMAL_MSG,$mysql_size,$SQL_CONN_RATIO,$MySQLstatus,$qsub,$qreu,$qcw,$q1,$q2,$q3,$q4,$q5,$q6,$q7,$dsmsc,$daim,$dmnp,$dreu,$dreua,$dsyncdb,$djsync,$djsyncc,$djagtx" >>
```

```
/home/temp/geo/gathered_data.csv
```

Appendix B

Sample Model Source Code

```
...
class WekaClassifier {

    public static double classify(Object[] i)
        throws Exception {

        double p = Double.NaN;
        p = WekaClassifier.N1dd3380(i);
        return p;
    }
    static double N1dd3380(Object []i) {
        double p = Double.NaN;
        if (i[1] == null) {
            p = 0;
        } else if (((Double) i[1]).doubleValue() <= 4.0) {
            p = WekaClassifier.N1c6ea391(i);
        } else if (((Double) i[1]).doubleValue() > 4.0) {
            p = WekaClassifier.Nb61ba312(i);
        }
        return p;
    }
    static double N1c6ea391(Object []i) {
        double p = Double.NaN;
        if (i[5] == null) {
            p = 0;
        } else if (((Double) i[5]).doubleValue() <= 1.0) {
            p = WekaClassifier.Nba96db2(i);
        } else if (((Double) i[5]).doubleValue() > 1.0) {
```

```

p = WekaClassifier.N2703405(i);
}
return p;
}
static double Nba96db2(Object []i) {
double p = Double.NaN;
if (i[2] == null) {
p = 0;
} else if (((Double) i[2]).doubleValue() <= 37.0) {
p = 0;
} else if (((Double) i[2]).doubleValue() > 37.0) {
p = WekaClassifier.N237363(i);
}
return p;
}
static double N237363(Object []i) {
double p = Double.NaN;
if (i[1] == null) {
p = 0;
} else if (((Double) i[1]).doubleValue() <= 3.0) {
p = 0;
} else if (((Double) i[1]).doubleValue() > 3.0) {
p = WekaClassifier.N34a91d4(i);
}
return p;
}
...

```

Bibliography

- [1] Anomaly Detection - Data Mining Techniques - Francesco Tamberi, Department of Computer Science, University of Pisa, 26 June 2007
- [2] Anomaly Detection: A Survey, Varun Chandola, Arindam Banerjee, and Vipin Kumar, August 15, 2007
- [3] Comparison of Unsupervised - Anomaly Detection Techniques - Bachelor Thesis - Author: Mennatallah Amer, Supervisor: Markus Goldstein, Reviewer: Prof. Dr. Andreas Dengel, Prof. Dr. Slim Abdennadher, Submission Date: 20 September, 2011
- [4] Edgeworth, F. Y. 1887. On discordant observations. *Philosophical Magazine* 23, 5, 364-375.
- [5] Anomaly Detection Lets You Find Patterns in Log Data, Thor Olavsrud, CIO, Sep 10, 2013
- [6] Kumar, V. 2005. Parallel and distributed computing for cybersecurity. *Distributed Systems Online*, IEEE 6, 10.
- [7] Spence, C., Parra, L., and Sajda, P. 2001. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*. IEEE Computer Society, Washington, DC, USA, 3.
- [8] Aleskerov, E., Freisleben, B., and Rao, B. 1997. Cardwatch: A neural network based database mining system for credit card fraud detection. In *Proceedings of IEEE Computational Intelligence for Financial Engineering*. 220-226.
- [9] Fujimaki, R., Yairi, T., and Machida, K. 2005. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM Press, New York, NY, USA, 401-410.
- [10] Statistical Techniques for Online Anomaly Detection in Data Centers - Chengwei Wang, Krishnamurthy Viswanathan, Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, Karsten Schwan - HP Laboratories HPL-2011-8
- [11] The Science of Anomaly Detection - How Grok Uses Machine Intelligence to Find Anomalies - White Paper
- [12] Tan, P.-N., Steinbach, M., and Kumar, V. 2005. *Introduction to Data Mining*. Addison-Wesley. Chapter 2.
- [13] Chandola, V., Banerjee, A., and Kumar, V. (2009), Anomaly detection : A survey. *ACM Computing Surveys (CSUR)*, 41, 15:1–58

- [14] Hodge, V. and Austin, J. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review* 22, 2, 85-126.
- [15] Agyemang, M., Barker, K., and Alhajj, R. 2006. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis* 10, 6, 521-538.
- [16] Markou, M. and Singh, S. 2003a. Novelty detection: a review-part 1: statistical approaches. *Signal Processing* 83, 12, 2481-2497.
- [17] Patcha, A. and Park, J.-M. 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Networks* 51, 12, 3448-3470.
- [18] Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., and Srivastava, J. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of SIAM International Conference on Data Mining*. SIAM.
- [19] Forrest, S., Warrender, C., and Pearlmuter, B. 1999. Detecting intrusions using system calls: Alternate data models. In *Proceedings of the 1999 IEEE ISRSP*. IEEE Computer Society, Washington, DC, USA, 133-145.
- [20] Snyder, D. 2001. Online intrusion detection using sequences of system calls. M.S. thesis, Department of Computer Science, Florida State University.
- [21] Dasgupta, D. and Nino, F. 2000. A comparison of negative and positive selection algorithms in novel pattern detection. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 1. Nashville, TN, 125-130.
- [22] A Survey of Outlier Detection Methods in Network Anomaly Identification, Prasanta Gogoi, D K Bhattacharyya, B Borah and Jugal K Kalita, revised 9 February 2011
- [23] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2), 2004.
- [24] M. Markou and S. Singh. Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83(12), 2003.
- [25] M. Markou and S. Singh. Novelty detection: A review - part 2: Neural network based approaches. *Signal Processing*, 83(12), 2003.
- [26] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2006. ACM Press.
- [27] A. B'anhalmi, A. Kocsor, and R. Busa-Fekete. Counterexample generation-based one-class classification. *Machine Learning: ECML 2007*, 2007.

- [28] W. Fan, M. Miller, S. J. Stolfo, W. Lee, and P. K. Chan. Using artificial anomalies to detect unknown and known network intrusions. In ICDM, 2001.
- [29] Anomaly Detection by Combining Decision Trees and Parametric Densities- Matthias Reif, Markus Goldstein, Armin Stahl - German Research Center for Artificial Intelligence (DFKI) - Thomas M. Breuel, Technical University of Kaiserslautern, Department of Computer Science
- [30] Bhawe, Chinmay, "BIG DATA CLASSIFICATION USING DECISION TREES ON THE CLOUD" (2013).Master's Projects. Paper 317.
- [31] Theiler, J. and Cai, D. M. 2003. Resampling approach for anomaly detection in multispectral images. In Proceedings of SPIE 5093, 230-240, Ed.
- [32] Abe, N., Zadrozny, B., and Langford, J. 2006. Outlier detection by active learning. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM Press, New York, NY, USA, 504-509.
- [33] Steinwart, I., Hush, D., and Scovel, C. 2005. A classification framework for anomaly detection. Journal of Machine Learning Research 6, 211-232.
- [34] V. Hodge and J. Austin. A survey of outlier detection methodologies. Artif. Intell. Rev., 22(2), 2004.
- [35] G. K. F. Tso and K. K. W. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," Energy, vol. 32, pp. 1761-1768, 2007.
- [36] D. Delen, G. Walker, and A. Kadam, "Predicting breast cancer survivability: A comparison of three data mining methods," Artificial Intelligent in Medicine, vol. 34, pp. 113-127, 2005
- [37] L. Y. Chang and W. C. Chen, "Data mining of tree-based models to analyze freeway accident frequency," Journal of Safety Research vol. 36, pp. 365-375, 2005.
- [38] Human talent prediction in HRM using C4.5 classification algorithm – Hamidah Jantan – IJCSE – Vol. 02, No. 08, 2010, page 2529.
- [39] Efficient C4.5 – Salvatore Ruggieri – IEEE Transactions on Knowledge and Data Engineering, Vol 14, No.2, March/April 2002 – p.438
- [40] Anomaly detection from server log data - A case study, Sami Nousiainen, Jorma Kilpi, Paula Silvonen & Mikko Hiirsalmi
- [41] Trettel, E.L. Forecasting Database Disk Space Requirements: A Poor Man’s Approach, Prepared for the CMG Conference Committee 32nd Annual International Conference of The Computer Measurement Group, Inc., December 4–9th 2006, Reno, Nevada.

- [42] Perng, C.-S., Ma, S., Lin, S. & Thoenen, D. Data-driven Monitoring Design of Service Level and Resource Utilization. 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005. 15–19 May 2005.
- [43] Knobbe, A., Van der Wallen, D. & Lewis, L. Experiments with data mining in enterprise management. In: Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 1999. Distributed Management for the Networked Millennium.
- [44] Yamanishi, K. & Maruyama, Y. Dynamic syslog mining for network failure monitoring. International Conference on Knowledge Discovery and Data Mining. Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, Chicago, Illinois, USA. Industry/government track paper, 2005. Pp. 499–508.
- [45] Ziv, J. On Classification with Empirically Observed Statistics and Universal Data Compression, IEEE Transactions on Information Theory, Vol. 34, No. 2, March 1988.
- [46] Fast Threat Detection with Big Data Security Business Intelligence, IT@Intel Brief, Intel IT, Risk Management, July 2013
- [47] ANOMALY DETECTION IN THE DATA CENTER AND THE CLOUD, May 2013, metaphor software
- [48] Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Chapter 8, Ian H. Witten, Eibe Frank, 2000