FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

# Web Usage Mining for Click Fraud Detection

**André Pacheco Pereira Neves**

Master in Informatics and Computing Engineering

Supervisor: José Luís Borges (Ph.D)

Co-Supervisor: Rosaldo Rossetti (Ph.D)

$14^{th}$ July, 2010

# Web Usage Mining for Click Fraud Detection

**André Pacheco Pereira Neves**

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Luis Paulo Gonçalves dos Reis (Ph.D)
External Examiner: Paulo Alexandre Ribeiro Cortez (Ph.D)
Supervisor: José Luis Cabral Moura Borges (Ph.D)

---

$27^{th}$ July, 2010

# Abstract

Online advertising is the main source of revenue for most Web sites, and is a business that moves millions of dollars. However, it soon became a target of fraudulent activities, where the most common type of such activities is known as click fraud.

AuditMark is a company that provides web auditing services for online advertising campaigns. This company gives their clients detailed reports on the quality of their web site's traffic originated from ad campaigns.

The main motivation behind this dissertation is to tackle the click fraud problem, which affects more and more companies. Several approaches to detect click fraud have already been proposed. Even so, these are not capable of detecting click fraud techniques that are likely to cause deviations in normal traffic patterns. The goal of this dissertation is to propose a new click fraud detection approach that detects these click fraud techniques, exploring Web Usage Mining methods. The result consists of a proof-of-concept developed to prove the applicability of these techniques in this specific domain. The proposed approach is able to detect these click fraud techniques, and follows a methodology based on unsupervised methods commonly employed in fraud detection scenarios.

In this approach, the use of synthetic data was required for the validation process. The used generated dataset includes both valid and invalid traffic, where the former was created to mimic the behavior of common click fraud techniques. The dataset contains the clickstream data over a time period. This time period is analyzed using a statistical analysis to model the base distribution (normal traffic pattern) and then detects deviations (outliers) from this distribution. These deviations are analyzed to characterize any possible click fraud attacks. This information is used to differentiate the clicks that caused the deviation from those that belong to the base distribution, using clustering techniques. Each clustering run results in a chosen cluster containing the suspicious clicks. For each cluster that is selected as being suspect of an attack, a suspicion score is given to all its elements. The results are then evaluated using a set of metrics commonly used in fraud detection problems.

By analyzing the experimental results, it is considered that this methodological approach is valid, accurate and robust. Even though the tests were conducted in a simulated environment, they demonstrated that the methodological steps are capable of detecting click fraud techniques that cause deviations in normal traffic patterns. Therefore, the main goal is accomplished. Web Usage Mining techniques were succesfully employed to the click fraud detection problem, and proved to be extremely useful in this context. A functional prototype was developed, which serves as proof-of-concept. However, only a small portion of the full potential of Web Usage Mining was explored, and much more work can be done in this specific area.

# Resumo

O mercado da publicidade online é a principal fonte de lucros da maioria dos Web sites, e é um negócio que movimenta milhões de dólares. No entanto, tornou-se alvo de actividades fraudulentas, cujo tipo mais frequente é conhecido como click fraud.

A AuditMark é uma empresa que fornece serviços de auditoria Web para campanhas publicitárias online, disponibilizando aos seus clientes relatórios detalhados acerca da qualidade do tráfego dos seus Web sites, proveniente de campanhas publicitárias.

A principal motivação desta dissertação é combater o problema da click fraud, que afecta cada vez mais empresas. Diversas abordagens foram propostas para detectar click fraud, mas nenhuma é capaz de detectar técnicas de click fraud que causam desvios nos padrões de tráfego normais. O objectivo desta dissertação é propor uma nova abordagem de detecção de click fraud que detecte estes tipos de click fraud, explorando técnicas de Web Usage Mining. O resultado é uma prova-de-conceito desenvolvida para demonstrar a aplicabilidade destas técnicas neste domínio específico. A abordagem proposta é capaz de detectar estes tipos de click fraud, seeguindo uma metodologia baseada em métodos não supervisionados frequentemente aplicados en cenários de detecção de fraude.

Nesta abordagem, foi necessário o uso de dados sintéticos para o processo de validação. O dataset gerado utilizado inclui tráfego válido e inválido, onde o último foi desenhado para imitar o comportamento de técnicas comuns de click fraud. O dataset contém dados de clicks efectuados durante um período de tempo. Este período de tempo é analisado estatisticamente para modelar a distribuição base (padrão de tráfego normal) e detectar desvios (outliers) desta mesma distribuição. Estes desvios são analizados para caracterizar possíveis ataques de click fraud, e esta informação é usada para diferenciar os clicks que causaram o desvio daqueles que pertencem à distribuição base, usando técnicas de clustering. De cada vez que o clustering é executado, um cluster é escolhido como sendo suspeito de conter um ataque e um score de desconfiança é atribuído a todos os seus elementos. Os resultados são então avaliados usando um conjunto de métricas normalmente utilizados em problemas de detecção de fraude.

Analizando os resultados experimentais, a abordagem metodológica é considerada válida, precisa e robusta. Apesar dos testes terem sido executados num ambiente simulado, estes demonstram que os passos metodológicos são capazes de detectar técnicas de click fraud que causam desvios nos padrões de tráfego normais. Assim sendo, o objectivo desta dissertação foi cumprido. Técnicas de Web Usage Mining foram empregues com sucesso para o problema que é a click fraud, e provaram ser extremamente úteis neste contexto. Um prótótipo funcional foi desenvoldido, que serve de prova-de-conceito. No entanto, apenas foi explorada uma pequena parte do potencial do Web Usage Mining, e ainda há muito trabalho que pode ser feito nesta área específica.

*To*
*my Family*
*and Friends*

# Acknowledgements

I would like to thank my supervisor, Dr. José Borges, who was always available for pointing me in the right direction and for bringing order to my sometimes chaotic ideas.

I am also grateful to Dr. Rosaldo Rossetti, my co-supervisor, for giving me very useful insights during the dissertation.

I would like to thank Mr. Pedro Fortuna and the people at AuditMark for helping me many times during my research, and for always providing me what i needed.

Certainly, without the strong support of my family and friends, i would not have been able to make this accomplishment. Their love and faith is my invaluable asset worth cherishing lifelong.

André Pacheco Pereira Neves

# Contents

# List of Figures

LIST OF FIGURES

# List of Tables

# Abbreviations

| | |
|---|---|
| ACC | Accuracy |
| AFS | Average Fraud Score |
| AMOC | Activity Monitoring Operating Characteristic |
| AUC | Area Under receiver operating Curve |
| AVS | Average Valid Score |
| AS | Analysis Subset |
| CART | Classification And Regression Tree |
| CB | Customer Behavior |
| CBR | Case-Based Reasoning |
| CEO | Chief Executive Officer |
| COI | Communities of Interest |
| CPA | Cost-Per-Action |
| CPC | Cost-Per-Click |
| CPM | Cost-Per-Mille |
| CSP | Continuous Sequential Patterns |
| CSST | Cluster Size Similarity Threshold |
| CST | Cluster Size Similarity |
| CXE | Cross Entropy |
| DBSCAN | Density-Based Spatial Clustering of Applications with Noise |
| DCP | Data Collection Plug-in |
| DDoS | Distributed Denial-of-Service |
| DPI | Data Processing Interface |
| DPP | Data Processing Plug-in |
| DST | Dimension Similarity Threshold |
| EDA | Exploratory Data Analysis |
| EM | Expectation Maximization |
| FN | False Negative |
| FNR | False Negative Rate |
| FP | False Positive |
| FPR | False Positive Rate |
| GEO | Geometrical Methods |
| HMM | Hidden Markov Model |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol (address) |
| JIC | JavaScript Interaction Code |

| | |
|---|---|
| k-NN | k Nearest Neighbor |
| LRS | Longest Repeating Subsequences |
| MAD | Median of the Absolute Deviation about the median |
| MINT | Mining Named-entity Transliteration equivalents |
| NO | Nominal Values |
| NU | Numerical Values |
| OLAP | Online Analytical Processing |
| OS | Operative System |
| PHP | Hypertext Preprocessor (recursive acronym) |
| PC | Personal Computer |
| PLSA | Probabilistic Latent Semantic Analysis |
| PPC | Pay-Per-Click |
| PR | Personalization |
| RBF | Radial Basis Functions |
| ROC | Receiver Operating Characteristic |
| ROI | Return over Investment |
| Q1 | $1^{st}$ Trimester |
| SEO | Search Engine Optimization |
| SI | System Improvement |
| SM | Site Modification |
| SOM | Self Organizing Maps |
| SP | Sequential Patterns |
| SQL | Structured Query Language |
| ST | Statistical Methods |
| SVM | Support Vector Machines |
| TN | True Negative |
| TNR | True Negative Rate |
| TP | True Positive |
| TPR | True Positive Rate |
| TO | Traffic Outlier |
| TU | Time Unit |
| TW | Time Window |
| UC | Usage Characterization |
| URL | Uniform Resource Locator |
| WUM | Web Utilization Miner |
| WWW | *World Wide Web* |
| ZC | modified Z-Test Confidence |

# Chapter 1

# Introduction

Currently the Internet is a way of communication that companies increasingly use as part of their marketing and advertising strategies. Online advertising is the main source of revenue for most websites. Giants such as Google, Yahoo and Microsoft are currently considered the main online advertising agencies, and move millions of dollars in this huge business. However, due to the enormous amounts of money involved it soon became a target of fraudulent activities. Probably the most common type of such activities is known as click fraud. According to Google, click fraud are "any clicks or impressions that may artificially inflate an advertiser's costs or a publisher's earnings". These are also known as "invalid clicks", and cause great losses to advertisers. Since the main business model is pay-per-click (PPC), advertisers pay for each click that is made on their ads. This means every click has to be paid, whether it's legitimate (valid) or not.

AuditMark is a company that provides web auditing services for online advertising campaigns. This company gives their clients detailed reports on the quality of their web site's traffic originated from ad campaigns. This information allows its customers (i.e. advertisers) to manage their advertising campaign the best way.

## 1.1 Motivation

The main motivation behind this dissertation is to tackle the click fraud problem, which affects more and more companies. It is responsible for great losses in online marketing campaigns and companies search desperately for solutions. Several approaches to detect click fraud have already been proposed. In [GGJR06, PZCG10], methods for fighting malicious scripts responsible for invalid clicks are proposed. [DSTT07, SGJ07, CL07] performed studies on botnets, a common tool used by fraudsters. Other suggestions and detection systems to address the click fraud problem can be found in [MKR07, JSJ07,

KWW+08, ZG08]. The problem is complex and undergoes constant evolution. Fraudsters are constantly inventing new ways of bypassing existing countermeasures, and detection systems must constantly be improved to detect new types of fraud. Thus, the difficulty of solving this problem increases as the sophistication level of new fraud methods increase. Developing an approach that successfully detects all kinds of attacks is destined to failure. The problem must then be solved using a "divide and conquer" strategy. By combining different techniques specialized in detecting different types of click fraud, the probability of detecting a larger portion of fraud is greater.

## 1.2 Aim and Goals

The aim of this dissertation is to propose a new click fraud detection approach, exploring Web Usage Mining techniques. The result is as a proof-of-concept developed to prove the applicability of these techniques in this specific domain. This approach is able to detect some types of click fraud attacks that cause a deviation in the underlying traffic patterns of the analyzed web site. These types of attacks are the ones that cause greater losses, because of the larger number of performed clicks in each attack. Examples are botnet attacks (also related to Distributed Denial-of-Service(DDoS) attacks), click farms and affiliated networks of people paid to perform click fraud.

AuditMark is able to analyze its customers' (advertisers) web traffic by using the AuditService auditing platform. To do so, a JavaScript call to the AuditService data collection module is inserted in the customer's website code. This call is inserted on the landing pages (pages to where the ads point to), and triggers the data collection process. Then, for each click made on the customer ads, the information about the visitor is gathered to be analyzed later. This information allows the characterization of the visitor who performed each click (e.g. operative system, Web browser, geographic location, user language and screen resolution). The analysis of this information results in the attribution of a score to each click. The score is then used to measure the analyzed website's traffic quality. So far, the clicks are analyzed individually and the context relative to the underlying traffic patterns is ignored. This thesis proposes to incorporate the usage of Web Usage Mining techniques into the AuditService platform. By analyzing the traffic patterns of the website, these techniques allow the detection of deviations from the normal traffic patterns. These deviations consist of sets of clicks, that when analyzed individually would be completely valid. However, when analyzed considering the website's normal traffic patterns they reflect a deviation which can be considered a potential organized attack. These organized types of click fraud are not detectable using the current techniques used by the AuditService.

The final result of the present work is a functional prototype for an automated analysis software tool, to be integrated in the AuditService afterwards. This software tool will be

added to the existing family of data processing plugins of the platform, where each one explores a different analysis approach, is fully independent and contributes to the final traffic quality score.

The general goal of this dissertation research is, in a nutshell, to develop a solution that detects click fraud attacks which cause deviations in normal traffic patterns of a website, exploring Web Usage Mining techniques.

## 1.3 Methodological Approach

In this approach, the use of synthetic data was required for the validation process. In a click fraud detection scenario, proving the validity of a given click depends mostly on the context and is sometimes impossible. When analyzing real web traffic data, there is never absolute certainty that a click is in fact invalid or valid. Also, it is not possible to hire fraudsters to commit fraud on a given web site and then report back, the doubt remains. Using synthetic data it is possible to compare the result of the analysis with the correct answer known apriori to be correct. Therefore, to evaluate the performance of the proposed solution, a generated dataset was used as input for the analysis. The generated dataset includes both valid and invalid traffic, where the former was created to mimic the behavior of common click fraud techniques. These include single person attacks, click farms, affiliated networks, automated-click tools and botnets.

The generated dataset is created using a traffic simulator developed for this purpose. A set of clicks is generated between two dates using stochastic methods, and a simulation log file is outputted. This log file is then used to perform the clicks, simulating a real scenario. A web page is designed to emulate several ads, which lead to the landing page containing the call to the AuditService collection module. The data relative to each click is then collected by the AuditService collection module and saved for posterior analysis. This data contains some additional information about the type of click (valid or invalid), for validation purposes. Some preprocessing is made on this generated dataset, which leaves it ready for the first step of the analysis.

The data between the two dates used for the traffic generation constitutes then the analysis time period for the remaining process. This time period is then split into several time windows, which are then overlapped. By overlapping the windows it becomes possible to analyze the traffic patterns that repeat over time. These time windows are split once more into smaller time slices called time units, to enable the comparison between equivalent periods across all time windows. Analysis subsets are then formed by grouping together each time unit from all time windows. Consider that the analysis time period was one month (e.g 30 days), the time window was set to one day, and the time unit was set to one hour. This analysis time period data would be divided into 30 different time windows (days), and then each time unit would be divided into 24 time units (hours). An example

of an analysis subset could be the 30 time units corresponding to the 13:00 to 14:00 (1 hour), from all the time windows (days). One can see that each of these analysis subsets will contain the traffic pattern during each hour (time unit), for every day (time window) of the month (analysis period).

Once the analysis period is divided into these analysis subsets, a statistical analysis models the base distribution (normal traffic pattern) and then detects deviations (outliers) from this distribution. Given that the base distribution is not know apriori, it must be modeled using the data itself. Basing the statistical analysis on the central limit theorem, the base distribution of each analysis subset is considered to be approximately normal. This enabled the usage of a statistical test designed for normal data, the modified z-score test. In a modified z-score test the z-score is determined based on outlier resistant estimators. The median and the median of absolute deviation about the median (MAD) are such estimators. Using these estimators, the median is considered to be the mean value that characterizes the base distribution. The modified z-score test is then applied and the outliers are labeled when they have a z-score greater than a defined threshold. This outlier detection method is applied in several dimensions of the data: absolute number of clicks, number of clicks per os, number of clicks per web browser, number of clicks per country and number of clicks per referrer. The results are deviations from normal traffic patterns, each one in a specific point in time of the analysis period, with specific characteristics. These deviations are now considered suspicious attacks of click fraud.

After the deviations have been detected in all time units and all dimensions, they are analyzed to characterize the suspicious attacks. This analysis groups dimensions that have a similar amount of deviation when compared to the base distribution. By doing so, only those deviations that really characterize the possible attack are grouped. The result of this attack characterization is a set of pairs attribute=value, where the attribute is the dimension, and the value the category of the dimension responsible for the deviation. It also contains an estimate of the amount of clicks that caused the deviation (possible attack size). Once all deviations are analyzed and the possible attacks characterized, this information is used to try and separate the clicks that caused the deviation from those that belong to the base distribution.

To differentiate the clicks that caused the deviation from those that belong to the distribution, a clustering algorithm is used. First, the complete data corresponding to each deviation time period (the time unit of the time window) is retrieved from the database and preprocessed. Each deviation data forms a dataset. Then, using each deviation analysis result, the clustering algorithm is configured accordingly and is applied to the respective dataset. From the resulting cluster set, a cluster is selected if it is sufficiently similar to the suspicious attack characteristics. This is done by comparing each cluster size with the estimated deviation amount, and its attribute values with the attack characteristics (pairs attribute=value).

For each cluster that is selected as being suspect of an attack, a suspicion score is given to all its elements. This score can me seen as a measure of traffic quality expressed in percentage, where 100% corresponds to a completely valid click, and 0% to an undeniable case of click fraud. The current score formula is quite simple and is calculated based on the similarity between the selected cluster size and the estimated attack size. If the cluster size is less or equal than the estimate size it is considered to contain the entire attack and a score of 0% is given. On the contrary, if the cluster size is greater than the estimate it is considered to contain false positives (valid clicks detected as invalid) and the score is computed as score = estimated size / cluster size. For instance, if the estimated attack size was 50 and the cluster had 100 elements, the resulting score would be 50%. Thus, this reflects the "certainty" that the solution has that those clicks are in fact invalid. Even though this scoring formula is extremely simple, it proved to be quite robust during the experiments as can be seen in Chapter 4.

When the detection process ends, the result is a set of clicks marked as being suspect of fraud, each one with a suspicion score. The final step of the methodology lies in validating these results. As mentioned above, the type (valid or invalid) of each click is already contained in the data. Analyzing this set of clicks, one can know the exact amount of: invalid clicks correctly marked as invalid (true positives); valid clicks incorrectly marked as invalid (false positives); valid clicks correctly marked as valid (true negatives) and invalid clicks correctly marked as valid (false negatives).

These amounts enable the calculation of performance metrics and the validation of the detection process. Considering that in a real life scenario one wouldn't have the information about the type of each click, the suspicion scores are also used as a performance metric. Since the suspicion scores are only altered for the the clicks marked as being invalid, the average scores for both true positives and false positives are computed and evaluated. All the clicks that are not marked have 100% scores, and would not bring any information for this analysis.

The steps described here summarize the methodological approach proposed in this dissertation. A complete description of the methodology can be found in Chapter 3.

## 1.4 Expected Contributions

This dissertation proposes to study a given website's underlying traffic patterns, from a statistical perspective. The normal traffic patterns of the website are modeled and abnormal deviations are detected. The detection of these deviations allows the usage of clustering methods that mark those clicks that are likely to have caused the deviations. These abnormal deviations in a website's traffic patterns may be the reflection of fraudulent activity. Without looking at these patterns, these fraudulent activities could pass unnoticed and cause severe losses to the advertisers who own the websites.

The contributions of the dissertation, can be summarized as follows:

1 - Propose a novel approach enabling the detection of some types of click fraud previously undetectable by current methods. To do so, the normal traffic patterns of the website are modeled and then deviations are detected. These deviations are then analyzed to characterize possible click fraud attacks. Together with these characterizations, the clicks contained in the deviation time period are clustered in order to differentiate the clicks that caused the deviation from those that belonged to the base traffic pattern. Once the clicks are differentiated they are given a suspicion score according to the system's certainty that they in fact contain a click fraud attack.

2 - The development of a robust traffic simulator to generate datasets including both valid and invalid traffic. All traffic is generated stochastically, and the distributions are completely configurable. Valid traffic is generated using statistical distributions observed in real traffic, and invalid traffic is created to mimic the behavior of common click fraud techniques. It can be seen as a click fraud penetration testing system, having some of the properties recommended in [KTP+09]. These are Isolation (does not influence real advertisers, publishers, and users), Resource Emulation (provide virtual resources to conduct more powerful attacks), Virtual Time Compression (speed up simulation time of attack) and Traffic Composition (complete control over the traffic).

## 1.5 Structure of this Dissertation

In Chapter 2, it is provided a background review of the related work in generic fraud detection, click fraud detection, Web usage mining and AuditMark's auditing platform, AuditService. Chapter 3 presents the proposed methodology for detecting some types of click fraud. Chapter 3 also contains some implementation details of the solution, and describes the traffic simulator developed to generate the used dataset. It also includes the evaluation methodologies for assessing the performance of the solution. The results obtained from the experiments conducted to test and validate the approach are presented in Chapter 4. Concluding remarks and extension are presented in Chapter 5.

# Chapter 2

# Related Work

The aim of this dissertation is to propose a new Click Fraud detection approach, exploring Web Usage Mining techniques. Online advertising is the main source of revenue for most websites, and the problem known as Click Fraud causes major losses. Web Usage Mining techniques capture patterns relative to the web users. Therefore, they are capable of detecting those patterns that characterize fraudulent behaviour in the online advertising context. By analyzing the advertiser's web traffic click-stream data, these techniques present a great potential for fighting Click Fraud.

This dissertation was inserted in the context of a company named AuditMark. AuditMark is a company that provides web auditing services for online advertising campaigns. To provide such services, the advertiser's web traffic is analyzed using the AuditService auditing platform. The goal of this dissertation is to create a Click Fraud detection software module exporing Web Usage Mining techniques. This software module will be integrated in the AuditService platform. Thus, a brief description of the AuditService platform and its architecture is described in Section 2.1. Given this is a Fraud Detection problem, Section 2.2 presents a detailed study on this subject. It contains commonly used techniques for fraud detection, major types of fraud and commonly used metrics. Section 2.3 presents a detailed review of the Click Fraud problem, as well as some existing approaches. A complete review of the state-of-the-art in Web Usage Mining is presented in Section 2.4. This review contains the commonly used methodology, techniques and applications of Web Usage Mining. Section 4.8 presents some final remarks about the study of the related work.

## 2.1 AuditMark's AuditService

AuditMark is a company that provides web auditing services for online advertising campaigns. This company gives their clients detailed reports on the quality of their web site's

traffic originated from ad campaigns. This information allows its customers (i.e. advertisers) to manage their advertising campaign the best way.

The AuditService is the platform that enables auditing the advertiser's web-traffic. By analyzing each web site's click-stream data, the AuditService outputs detailed reports on the quality of the audited traffic. The process to do so starts in data gathering and storage, passes thru a data processing phase and ends in result visualization. Due to the amount of data gathered, special attention was given to the architecture.

This section briefly describes the AuditService. The level of detail is kept to a minimum, to at the sime allow the reader a good perspective of the context, without compromising any proprietary information.

### 2.1.1 Architecture

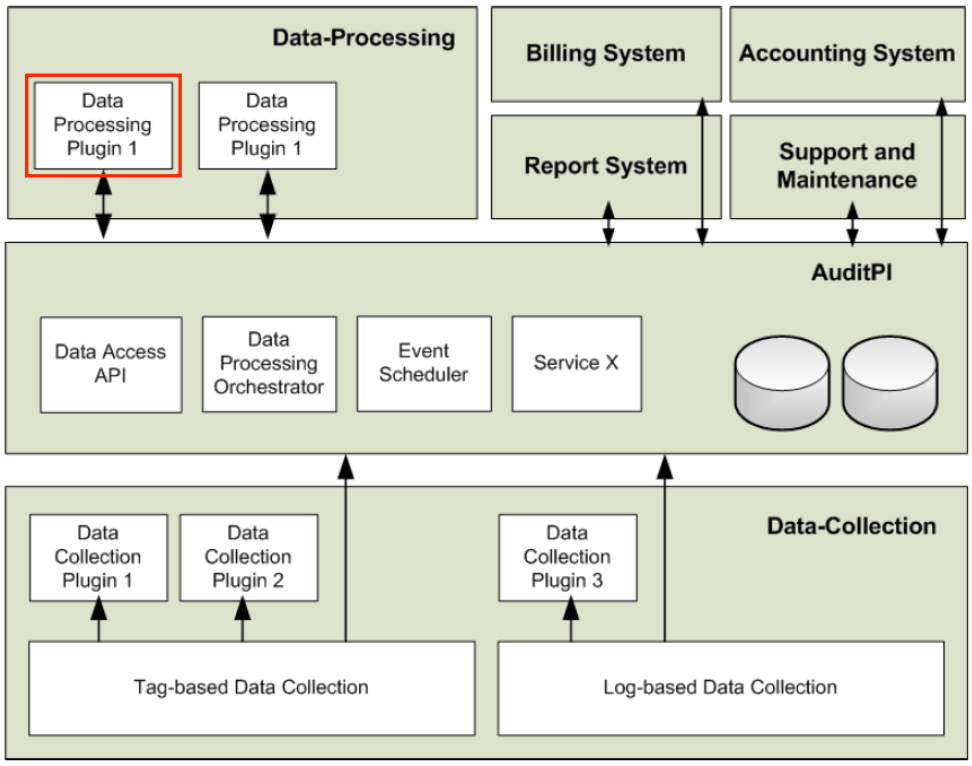


Figure 2.1: AuditService's Architecture Diagram

The AuditService has a layer-based architecture with three layers, as can be seen in Figure 2.1. The module marked in red is where the solution will be deployed. Each layer can be summarized as follows:

- **AuditPI Client Applications Layer**: This layer contains applications that use the AuditPI Layer's services. Exaples of AuditPI clients can be Data-Processing Plug-ins, and business related applications such as Billing and Reporting.

- **AuditPI Layer**: the AuditPI is the service framework. It is the most important part of the service and includes functions such as transparent distributed database access, data processing orchestration, message queuing system, event scheduler, etc.

- **Data-Collection Layer**: includes one or more components that implement web-traffic data collection methods. The two main data collection methods are Tag-based Data Collection and Log-based Data Collection. The former requires embedding in the customer's web page a JavaScript call to the AuditService. The latter can be done offline. The customer only has to send the server log file.

  To assure scalability, effectiveness and efficiency the service is run in a cluster where each machine runs an instance of the Data-Collection, an instance of the AuditPI Framework and a given number of AuditPI clients. Special attention is given to the AuditPI Client Application Layer and the Data Collection Layer. In the first layer is where the actual detection process takes place. The Data Collection Layer is where the data is collected using the Tag-Based Collection module. The main component of the Tag-Based Collection module is the Java Interaction Code, which is responsible for the collection of data using JavaScript interactions.

### 2.1.2 AuditPI Client Applications Layer

As referred before, the AuditPI Client Applications Layer consists of either Data-Processing plug-ins or Business related applications. These plug-ins can be just to process data, or they can also be responsible for performing other administrative tasks such as billing, accounting, maintenance, and support. To our problem we will consider only the Data-Processing plug-ins. These can be either auditing plug-ins or processing support plug-ins. The first are the ones who perform the real auditing and the latter aid the others in a given task.

#### 2.1.2.1 Data Processing Plug-in (DPP)

Given this, a Data Processing Plug-in (DPP) can be defined as the component responsible for processing data and performing the necessary auditing in the AuditService framework. Each DPP runs independently from the others, and explores a different click fraud detection approach. For each set of clicks being analyzed, each DPP produces a traffic quality score. The scores from all DPP's are then used to produce the final score for each click. This gives the system more robustness. The detection errors (wrong scores) of each DPP are attenuated by the scores from the remaining DPP's. The goal of this dissertation is to develop a DPP, that explores Web Usage Mining techniques. It must also contribute with it's own independent traffic quality score.

### 2.1.3 Data-Collection Layer

As its name implies, this layer is responsible for collecting the data to be used by the remaining modules of the application. It has two data sources, namely Tag-Based Data and Log-Based Data, as said before. It then hosts a series of Data-Collection plug-ins that operate on this data, for the most distinct tasks. Currently the Log-Based Data Collection module is not being used. The reason is that there is a possibility that the advertiser (client) itself can manipulate the logs. By modifying its own logs an advertiser could "create" invalid traffic, to later claim some sort of refund to the ad provider. The Tag-Based Collection cannot be manipulated since the data is collected directly from each click and saved directly on the database.

#### 2.1.3.1 Tag-Based Collection and AuditService's JavaScript Interaction Code (JIC)

Since the Log-based Data Collection module is deactivated, the only source of data is the Tag-based Data Collection. This component is responsible for capturing the HTTP requests of the Tag-Based interaction between the AuditService Server and the remote browser that's accessing the client's website. It acts as a pseudo-proxy in order to avoid the data from being manipulated. This ensures that the collected data corresponds exactly to reality. The main technology behind this module is known as JavaScript Interaction Code (JIC). This is a mechanism that collects click information to assist auditing. It's called JIC because it is an interaction between the user's browser and AuditService, co-ordinated by JavaScript code. The code is executed in rounds, where each round collects some information and triggers the next round. In the first round a unique id is created and some information is retrieved using JavaScript. In the end of the first round the script checks if the client has flash support and if so a flash object is also executed on the client. This will allow retrieving more information on the 2nd round, besides the one gathered by JavaScript. This process is highly dynamic, with each JS round being dynamically generated using PHP.

As JavaScript runs on the visitor's side, JIC includes a number of obfuscation techniques to difficult reverse engineering of the code and therefore any countermeasures to adulterate the information sent. The audited page must include a call to the 1st Round script where the interaction is triggered. At least one call to the script has to be made on the index page (current situation), but ideally all the web site pages should contain a JIC call. If every page contained a JIC call, valuable information about each user's session would be collected. This way other techniques could be applied in order to obtain richer patterns, such as sequential and navigational patterns.

#### 2.1.3.2 Data collected by the AuditService (JIC)

JIC uses different methods to extract as many information as possible from the web page visitors. These methods include JavaScript, Flash(when available) and Java. Using JavaScript several HTML DOM Objects can be accessed. Each DOM object can have their properties retrieved, as well as their public methods invoked. These objects are for instance Navigator, Screen, Document and Window. The Navigator object contains properties relative to the user's browser. The retrieved properties are userAgent, appCodeName, appName, appVersion, platform, language, among others. The Screen object contains definitions about the user's screen. For instance, properties like height, width, colorDepth, pixelDepth are extracted. The Document object contains information relative to the HTML document loaded by the browser. Properties such as selectedStyleSheetSet, preferedStyleSheetSet, compatible, actualEncoding, documentReferrer are retrieved. The Window object represents an open window in a browser. Some of the properties extracted are frame, length, java and getSearchEngine. These properties have String data type. However, the final data has additional binary values for each property that are 0 when the string is null and 1 otherwise.

If the user's browser has Flash installed, it allows the extraction of several other attributes. Examples are userhasAudio, userhasPrinting, userOS, userScreenX, userScreenY, userDPI, useravHardwareDisable. The value types of these attributes can be String, Int or Boolean.

Finally, if the client has Java installed, thru the Java Virtual Machine (JVM) many more data about each visitor can be retrieved. These include Java system properties such as java.runtime.version, java.vendor, java.specification.version, os.name, os.version, http.agent, user.name, user.language, user.timezone, user.country, getHostAddress, getHostName among many others. The value types are normally of String type, but binary values are as well defined for some attributes in the same way as JavaScript properties, mentioned above. However, currently the JIC is not collecting any data thru the client's JVM. The main reason is that the JVM was too slow retrieving the information. This severely affected the page loading times, and therefore was removed until the JIC's Java module is optimized.

## 2.2 Fraud Detection

Fraud can be defined as the use of false representations to gain an unjust advantage. It is as old as humanity and can take an unlimited number of forms. Although, the recent developments in technology that enabled today's global communication network also allowed even more ways for criminals to commit fraud. Classical types of fraudulent activities such as money laundering became easier to commit and new forms of fraud emerged,

such as mobile telecommunications fraud, computer intrusion, and click fraud. There are normally two ways to fight fraud, fraud prevention and fraud detection. Fraud detection comes into play once fraud prevention has failed. However, fraud detection must always be used since the system is normally unaware that fraud prevention has failed. Given that fraud prevention measures are out of scope for this thesis, they will not be further discussed.

Fraud detection has been implemented using a large variety of techniques. It consists of identifying fraud as quickly as possible once it has occurred, which requires the detection module to be accurate and efficient. It is a constantly evolving field because whenever a detection method becomes known, criminals will adapt their strategies or create new ones. Even so, new criminals are always entering the field and will not be aware of the current fraud detection methods. Thus, they will tend to adopt "old" strategies that lead to identifiable frauds. This means that the earlier detection tools need to be applied as well as the latest developments. Yet, developing new fraud detection methods becomes more difficult because the exchange of ideas in fraud detection is severely limited. It makes perfect sense, considering that if these techniques were described in great detail in the public domain, criminals would have the information they need to evade detection. The data is usually highly skewed, which means that the observations relative to fraud are very little when compared to the remaining observations.

This chapter summarizes several recent statistical and data mining techniques for detecting fraud. The most known types of fraud and common metrics used in this field are also described.

### 2.2.1 Fraud Detection Methods

Statistical and data mining fraud detection methods can be supervised, unsupervised or hybrids. Supervised methods are commonly employed when samples for both fraudulent and non-fraudulent records exist. These samples are then used to construct models which later assign new observations into one of the two classes. Obviously, it is required a high level of confidence about the true classes of the original data. In addition, these models are only able to detect frauds of a type which have previously occurred.

Unsupervised methods usually search those records (customers, accounts, etc) which differ most from the norm. Alarms are generated so that these records, commonly known as outliers, can be examined more closely. Outliers can be seen as nonstandard observations, which can have many other causes other than fraud. Because of this, detected outliers cannot be directly marked as being fraud. Rather, the analysis should generate alerts when an anomalous observation has occurred, or is more likely to be fraudulent. This way it can be investigated in more detail, and avoid a potential detection error.

Furthermore, combinations of several methods are explored in the hope of improving overall results. These combinations are called hybrids, and combine methods of the same or different types (supervised/unsupervised). The main objective is to use each method's positive aspects while minimizing its negative aspects' influence on the final outcome.

#### 2.2.1.1 Supervised Methods

These techniques require knowledge of positively identified cases of fraud and legitimate actions (observations). This information is essential in order to build the models, which require the class to be provided to the learning phase of the algorithms. These models then produce a suspicion score for new cases. Examples of commonly used supervised learning algorithms are Neural networks, decision trees, rule induction and support vector machines (SVM). Some important considerations have to be made when building a supervised model for fraud detection. Uneven class sizes and different costs for different types of errors are two of these considerations. The costs of investigating observations and the benefits of identifying fraud must be also considered. Another common problem is that class membership may be uncertain. Fraudulent observations can be labelled as legitimate since they are not known and/or legitimate observations might be misreported as fraudulent.

A three-layer, feed-forward Radial Basis Function neural network was used by [GR94] to detect credit card transaction fraud. It only needed two training passes to produce a fraud score in every two hours for new transactions. [BKJ03] used a multi-layer neural network with exponential trace memory to detect temporal dependencies in synthetic Video-on-Demand log data. Fuzzy neural networks on parallel machines to speed up rule production for customer-specific credit card fraud detection were proposed in [SZP02]. A comparison study between the performance of a neural network and a bayesian network, for credit transactional fraud detection can be found in [MTVM02]. The used algorithms were STAGE for the bayesian network and back-propagation for the neural network. SVM ensembles with bagging and boosting with aggregation methods were proposed in [KOO03] for subscription fraud.

Decision trees, rule induction, and case-based reasoning have also been used. A systematic data selection to mine concept-drifting, possibly insufficient, credit card data streams was introduced in [Fan04]. With the same credit card data, [WFYH03] proposes a pruned classifier C4.5 ensemble which is derived by weighting each base classifier according to its expected benefits and then averaging their outputs. It is demonstrated that the ensemble will most likely perform better than a single classifier which uses exponential weighted average to emphasize more influence on recent data. [RMN+99] presents a two-stage rules-based fraud detection system. It starts by generating rules using a modified C4.5 algorithm and then sorts the rules based on accuracy of customer level rules to

select them based on coverage of fraud of customer rules and difference between behavioral level rules. It was applied to a telecommunications subscription fraud. A boosted C5.0 algorithm was used by [BGMP99] on tax declarations of companies. [SZC02] applied a variant of C4.5 for customs fraud detection. Case-based reasoning (CBR) was used by [WA00] to analyse the hardest cases which have been misclassified by existing methods and techniques. A 20% higher true positive and true negative rates than common algorithms on credit applications was claimed by the authors.

Traditional statistical modelling such as linear discriminant analysis and logistic discrimination [Han81, McL92] have proven to be effective tools. [FS01] use least squares regression and stepwise selection of predictors to show that standard statistical methods are competitive.

Other techniques include expert systems, association rules, and genetic programming. Expert systems have been applied to insurance fraud. [JM02] have implemented an actual five-layer expert system in which expert knowledge is integrated with statistical information assessment to identify medical insurance fraud. Experiments with fuzzy expert systems were made by [PVS05], [SG01] and [vA97]. [DT97] applied an expert system to management fraud. A Fraud Patterns Mining (FPM) algorithm, modified from Apriori, to mine a common format for fraud-only credit card data is presented by [CT04]. A genetic programming with fuzzy logic to create rules from classifying data was used by [Ben00]. This system was tested on real home insurance claims and credit card transaction data. However, none of these papers on expert systems, association rules, and genetic programming provide any comparison analysis with the many other available methods and techniques.

The above supervised algorithms are conventional learning techniques which are designed to only process structured data from single 1-to-1 data tables. Fraud detection can benefit from applying relational learning approaches such as Inductive Logic Programming (ILP) [MdR94] and simple homophily-based classifiers [PPPM03] on relational databases.

#### 2.2.1.2 Unsupervised Methods

These methods apply when there are no previous observations of fraudulent and legitimate cases. Combinations of profiling and outlier detection are the most common approaches to this kind of problem. A base distribution is modeled to represent normal behavior (or profile) to then detect the system attempts to detect observations that show the greatest departure from this norm.

Link analysis and graph mining are hot research topics in antiterrorism, law enforcement, and other security areas, but these techniques seem to be relatively under-rated in fraud detection research. In [CPV01] the temporal evolution of large dynamic graphs'

is studied for telecommunications fraud detection. Each graph is made up of subgraphs called Communities Of Interest (COI), which result of linking phone accounts using call quantity and duration. This resulted in the conclusion that fraudulent phone accounts were linked, because fraudsters call each other or the same numbers. Also, their behavior remains the same and is observable in new accounts.

Unsupervised neural networks have been applied in [BST01] for mobile telecommunications fraud detection. A recurrent neural network to form short-term and long-term statistical account behaviour profiles is used. Hellinger distance is used to compare the two probability distributions and give a suspicion score on telecommunications toll tickets. [DGSC97] introduce a non-linear discriminant analysis algorithm that does not require labels.

The most popular unsupervised method used in data mining is clustering. This technique is used to find natural groupings of observations in the data and is especially useful in market segmentation. However, cluster analysis is sensitive to the choice of metric (the way variables are scaled, transformed and combined to measure the similarity between observations). Observations may cluster differently on some subsets of variables than they do on others which may result in more than one valid clustering result in the same data set. Clustering allows the formation of local models from which we can find local outliers in the data. In the context of fraud detection, a global outlier is a transaction anomalous to the entire data set. Local outliers describe transactions that are anomalous when compared to subgroups of the data and thus are effective when the population is heterogeneous. Using medical insurance data, [YiTWM04] applied the unsupervised SmartSifter algorithm which can handle both categorical and continuous variables, and detected statistical outliers using Hellinger distance. [BBHH01] recommend Peer Group Analysis to monitor inter-account behaviour over time. It consists in comparing the cumulative mean weekly amount between a target account and other similar accounts (peer group) at subsequent time points. The distance metric/suspicion score is a t-statistic which determines the standardised distance from the centroid of the peer group. On credit card accounts, a time window of thirteen weeks to calculate peer group and a future time window of four weeks are used. To monitor intra-account behavior over time [BBHH01] also suggest Break Point Analysis. It detects rapid spending or sharp increases in weekly spending within a single account. Accounts are ranked by the t-test. The fixed-length moving transaction window contains twenty-four transactions: first twenty for training and next four for evaluation on credit card accounts.

[BBD+02] recommends Principal Component Analysis of RIDIT scores for rank-ordered categorical attributes on automobile insurance data. [HT98] introduce an experimental real-time fraud detection system based on a Hidden Markov Model (HMM).

#### 2.2.1.3 Hybrid Methods

Using a combination of methods enables exploring the strengths of each method while minimizing the limitations due to its weaknesses. These combinations can be supervised hybrids and supervised/unsupervised hybrids. Supervised hybrids are combinations of several supervised algorithms whereas supervised/unsupervised hybrids combine both approaches.

- Supervised Hybrids

  These approaches can be categorized into bagging, stacking and stacking-bagging. The main goal of these methodologies is to increase predictive performance when compared to a single model.

  **Bagging**: Bagging is based on combining the predictions of different algorithms. This is usually made using weights for each classifier, which can be the same for all (equal to the average) or customized to improve performance. These weighted predictions are then merged to produce the final classification for a given example.

  **Stacking**: Stacking normally consists of combining models from different types, thus built by different learning algorithms. Stacking introduces the concept of a meta-learner, which replaces the voting procedure used in bagging. The meta-learner algorithm tries to learn which base classifiers are the reliable ones, in order to discover the best way to combine their output. For example, [CFPS99] performed stacking by combining naive Bayes, C4.5, CART, and RIPPER as base classifiers. The results indicated high cost savings and better efficiency on credit card transaction fraud.

  **Stacking-Bagging**: Stacking-Bagging uses both previous techniques. It uses stacking to learn the relationship between classifier predictions and the correct class. The chosen base classifiers' predictions then contribute their individual votes and the class with the most votes is the final prediction. This approach was used in [PAL04], which used backpropagation neural networks, naive Bayes, and C4.5 as base classifiers on data partitions derived from minority oversampling with replacement. This approach produced best cost savings on automobile insurance claims.

- Supervised/Unsupervised Hybrids

  By using both types of methods together, one can hopefully benefit from the main advantages of the two. A lot of work using this combination has been done in telecommunications fraud detection. [CP01] proposed the use of daily (time-driven) account summaries which are then added to the training set and processed by supervised algorithms like atree, slipper and model-averaged regression. These summaries (signatures) which are assumed to be legitimate are used to detect significant

deviations in calling behavior. [CLPS02] assign an averaged suspicion score to each call (event-driven) based on its similarity to fraudulent signatures and dissimilarity to its account's normal signature. Calls with low scores are used to update the signature using exponential weights, where recent calls are weighted more heavily.

[FP97] generate fraud rules from each phone account's labelled data and select the rules that cover most accounts. To find anomalies, each selected fraud rule is applied in the form of monitors (number and duration of calls) to the daily legitimate usage of each account. These monitors' output and labels of an account's previous daily behavior are used as training data for a simple Linear Threshold Unit. if the suspicion score on the next evaluation day exceeds its threshold, an alarm is raised on that account.

Unsupervised approaches have been used to segment the insurance data into clusters for supervised approaches. [WH97] apply k-means for cluster detection, C4.5 for decision tree rule induction, and domain knowledge, statistical summaries and visualisation tools for rule evaluation. On medical insurance claims data, [Wil99] uses a genetic algorithm, instead of C4.5, to generate rules and to allow the domain user, such as a fraud specialist, to explore the rules and to allow them to evolve accordingly. For automobile injury claims, [PB98] present a similar methodology using Self Organising Maps (SOM) for cluster detection before backpropagation neural networks. [E.95] uses an unsupervised neural network followed by a neuro-fuzzy classification system to monitor medical providers' claims.

Unconventional hybrids were proposed by [HWGH97] and [BLH99]. The first used backpropagation neural networks, followed by SOMs to analyse the classification results on medical providers' claims. The latter RBF neural networks to check the results of association rules for credit card transactions.

In a study made by [MEH+99], it is shown that supervised neural network and rule induction algorithms outperform two forms of unsupervised neural networks which identify differences between short-term and long-term statistical account behaviour profiles. The best results are from a hybrid model which combines these four techniques using logistic regression. [THHT98] claim that supervised neural networks and Bayesian networks on labelled data achieve significantly better outcomes than unsupervised techniques such as Gaussian mixture models on each non-fraud user to detect anomalous phone calls.

### 2.2.2 Types of Fraud

As said before there are several types of fraud. The most common types are credit card frauds, telecommunication frauds, insurance frauds, internal fraud, computer intrusion,

web network fraud and customs frauds. These common types will be described briefly to give the reader a notion of the variety of fraud that exists in the present day. A more detailed review of these types of fraud can be found in [LA09] and [BHH02].

*Web Network Fraud*: Web network fraud is divided into two categories: web advertising and online auction fraud. Web advertising network fraud is a fraud in the Internet advertising business. The most common type is known as Click Fraud, which is described in detail in Section 2.3. Online auction fraud is a type of illicit activity where commonly fraudsters lie about the product they are auctioning or fail to deliver it after receiving the payment.

*Internal Fraud*: Companies sometimes are victims of fraud committed from within. This type of illicit activity is known as internal fraud. It normally consists of fraudulent financial reporting by management and abnormal retail transactions by employees.

*Insurance Fraud*: Insurance fraud affects the many types of insurances, and are usually divided into home insurance fraud, crop insurance fraud, medical insurance fraud and automobile insurance fraud.

*Customs Fraud*: Customs is how country administrate the imports and exports of goods, and collect the duties. These duties constitute one of the biggest sources of revenue of a country. Fraudsters usually hide merchandise, declare less or make false reports to avoid regulation or having to pay the duties.

*Credit Fraud*: This type of fraud is divided into two categories: credit application fraud and credit card transaction fraud. Application fraud is a manifestation of identify crime, where application forms contain synthetic or stolen identity information. Credit card transaction fraud consists of using other person's credit card for own benefit and is divided into two types: offline and online fraud. Offline fraud is committed by using a stolen physical card. Online fraud is committed usually thru the web or phone shopping. It occurs when only the card's details are needed, and not any manual signature or card imprint are required.

*Computer Intrusion*: Intrusions are sets of actions that attempt to compromise the integrity, confidentiality or availability of a resource. Common targets for intrusions are user accounts, file systems, system kernels or specific resources.

*Telecommunication Fraud*: The telecommunication industry is severely affected by fraud. There are many types of fraud and they occur at different levels, but the main types are subscription fraud and superimposed fraud. Subscription fraud consists of obtaining an account without intention to pay the bill. On the contrary, superimposed fraud is when a fraudster has taken over a legitimate account.

### 2.2.3 Metrics

Fraud detection can be seen as mapping instances into one of two classes (binary classification). In two-class prediction problem, outcomes are labeled either as belonging to positive (p) or negative (n) class. Thus, there are four possible outcomes from a fraud detection binary classification system. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP). However, if the actual value is n then it is said to be a false positive (FP). On the contrary, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p.

In the fraud detection context, positives are cases marked as fraud and negatives cases marked as being legitimate. Therefore, true positives are correctly detected cases of fraud whereas false positives are legitimate cases that are classified as fraud. True negatives are accurately detected legitimate cases, and false negatives are undetected cases of fraud.

The four outcomes can the be formulated as a contingency table, or confusion matrix.

Table 2.1: Binary classification contingency table (or confusion matrix)

| | | Actual Value | |
|---|---|---|---|
| | | p | n |
| Prediction | p' | True Positive (TP) | False Positive (FP) |
| | n' | False Negative (FN) | True Negative (TN) |
| | Total | P | N |

Some evaluation metrics that can be derived from this matrix are:

True positive rate (TPR) or sensitivity (also called recall):

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN} \tag{2.1}$$

False positive rate (FPR):

$$FPR = \frac{FP}{N} = \frac{FP}{FP+TN} \tag{2.2}$$

Accuracy (ACC):

$$ACC = \frac{TP+TN}{P+N} \tag{2.3}$$

True negative rate (TNR) or specificity:

$$TNR = \frac{TN}{N} = \frac{TN}{FP+TN} = 1-FPR \tag{2.4}$$

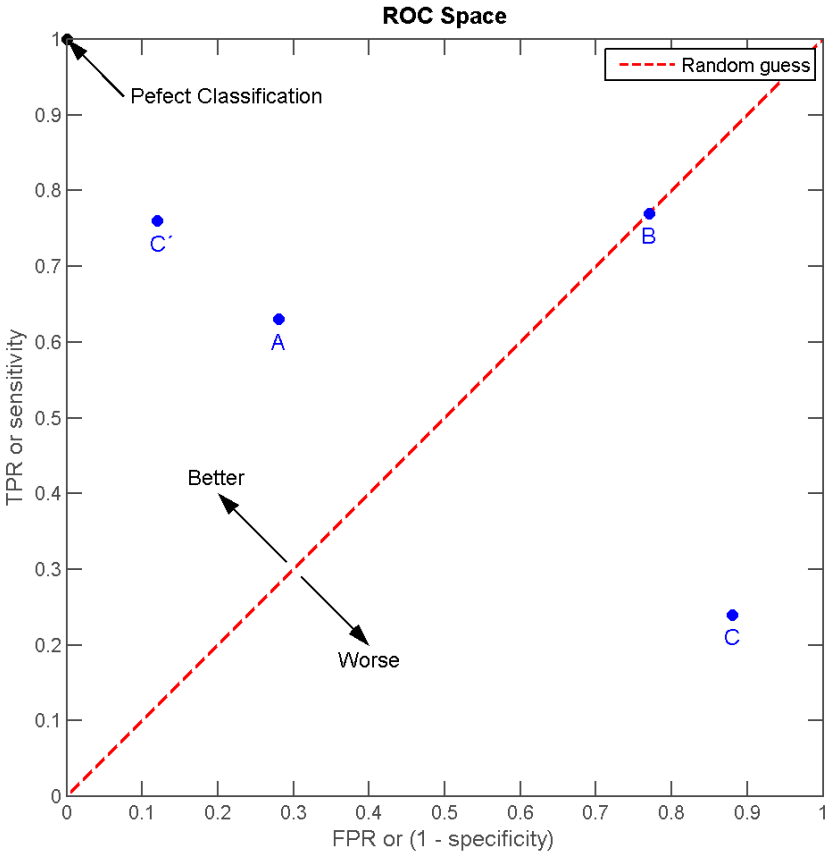


Figure 2.2: ROC space plot example, containing four examples

False negative rate (FNR):

$$FNR = \frac{FN}{N} = \frac{FN}{FN + TP} \tag{2.5}$$

A common way to visualize the performance of a binary classifier is to use a receiver operating characteristic (ROC) chart, or simply ROC curve. This is a graphical plot of the true positives (sensitivity) vs. false positives (1 - specificity), as its discrimination threshold is varied. Thus, to draw an ROC curve only the true positive rate (TPR) and false positive rate (FPR) are needed. TPR determines the performance on classifying positive instances correctly among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results occur among all negative samples available during the test.

A ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). Since TPR is

equivalent with sensitivity and FPR is equal to 1 - specificity, the ROC graph is sometimes called the sensitivity vs (1 - specificity) plot. Each prediction result or one instance of a confusion matrix represents one point in the ROC space.

The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). The (0,1) point is also called a perfect classification. A completely random guess would give a point along a diagonal line (the so-called line of no-discrimination) from the left bottom to the top right corners. An intuitive example of random guessing is a decision by flipping coins (head or tail). such a random guessing forms the diagonal that divides the ROC space. Points above the diagonal represent good classification results, points below the line poor results. An example of a ROC space plot can be seen in Figure 2.2, which illustrates several concepts described previously.

Examples of other proposed metrics are Area under the Receiver Operating Curve (AUC), Cross Entropy (CXE) and Activity Monitoring Operating Characteristic (AMOC). AUC measures how many times the instances have to be swapped with their neighbours when sorting data by predicted scores. CXE measures how close predicted scores are to target scores and AMOC analyzes average score versus false alarm rate (false positive rate). Many other metrics are employed, as will be seen in the following review.

Fraud departments commonly place monetary value on predictions to maximize cost savings/profit according to their policies. These can have the form of explicit costs [PAL04, CFPS99, FP97] or benefit models [Fan04, WFYH03]. [CLPS02] suggests giving a score for an instance (phone call) by determining the similarity of it to known fraud examples (fraud styles) divided by the dissimilarity of it to known legal examples (legitimate telecommunications account).

Many of the fraud detection studies using supervised algorithms since 2001 abandoned measurements such as true positive rate and accuracy at a chosen threshold (number of instances predicted correctly, divided by the total number of instances). The cause is that fraud detection misclassification costs (false positive and false negative error costs) are unequal, uncertain, can differ from example to example, and can change over time. In fraud detection, a false negative error is usually more costly than a false positive error. Although, some recent studies on credit card transactional fraud [CCHC04] and telecommunications superimposed fraud [KOO03] still aim to only maximise accuracy. Receiver Operating Characteristic (ROC) analysis is one of the metrics used.

From the studied publications on fraud detection, only [VDD04] sought to maximise Area under the Receiver Operating Curve (AUC) and minimise cross entropy (CXE). In addition, [VDD04] and [FS01] seek to minimise Brier score (mean squared error of predictions). [CNM04] use the average of mean squared error, accuracy, and AUC. It claims that the most effective way to assess supervised algorithms is to use one metric from threshold, ordering, and probability metrics. [FP99] state that Activity Monitoring

Operating Characteristic (AMOC) is suited for timely credit transactional and telecommunications superimposition fraud detection.

[LX01] propose entropy, conditional entropy, relative conditional entropy, information gain, and information cost for semi-supervised approaches, such as anomaly detection. For unsupervised algorithms, [YiTWM04] used the Hellinger and logarithmic scores to find statistical outliers for insurance. [BST01] employed Hellinger score to determine the difference between short-term and longterm profiles for the telecommunications account. [BBHH01] employ the t-statistic as a score to compute the standardised distance of the target account with centroid of the peer group; and also to detect large spending changes within accounts.

Other important measures include how fast the frauds can be detected (detection time/time to alarm), how many styles/types of fraud were detected, whether the detection was done in online/real time (event-driven) or batch mode (time-driven) [GR94]. There are also problem-domain specific criteria in fraud detection. It is common the involvement of domain experts and fraud specialists, and performing comparisons between automated and manual detection. Some types of visual analysis were also employed.

## 2.3 Click Fraud

Since the commercialization of the world-wide web (WWW) in the mid- 1990's, the marketplace has explored many alternatives in order to obtain revenues from online content. From online forms to gather information about customers and their desires, to online brochures, companies tried many techniques to publish their services and attract customers. These interactive brochures (commonly known as banners) can be considered as an early form of online advertising, and we can still find them on some older websites. In order to to help users find information, products, and services to help satisfy their needs, several directories (e.g., the Yahoo! directory) and keyword search engines (e.g. AltaVista) were developed to help direct users to destination sites. Companies could request or pay to be listed in online directories, and could optimize their web sites to show up prominently in search results provided by keyword search engines. Directories business model consisted in displaying banner ads that then linked to destination pages. These ads where determined according to user categories that were naively defined based on their browsing patterns, which turned out an unsuccessful targeting. On the other hand, search engines provide a much better way of targeting, using keyword search terms. Thru these keywords, the user is directly telling the search engine what (s)he is interested in, and the need of categorization based on browsing behavior becomes less necessary for a successful targeting. Other cause for search engines advertising success was that ads were displayed in textual form (as opposed to banner ads), resulting in a more targeted, less

intrusive user experience. [DSTT07] states that the combination of better targeting and better user experience made online advertising via search engines successful.

There are three major revenue models for online advertising: Cost Per Mille (CPM) - the advertiser pays per thousand impressions, i.e. ad displays; Cost Per Click (CPC) - the advertiser pays per click, i.e. when someone clicks its ads; Cost Per Action (CPA) - the advertiser is only charged when a determined action occurs (e.g. a sale) Two other common terms in online advertising are syndication and referral deals, which have their own way of providing revenues from this kind of advertisements. Syndication consists in an ad network syndicating their ads to other destination sites, which can be search engines or ordinary websites. This way, the ad network (which can be run by a search engine) provides ads to its syndicators and when these ads are clicked, the syndicator receives a share of the CPC revenues. Referral deals consist of a web site paying another for web traffic to itself. In a CPC referral deal, web site A pays web site B a referral fee every time a user clicks on a link on web site B, leading to A.

Online advertising is today's biggest source of revenues for both search engines and ordinary websites. Internet advertising revenue grew 15% during the 1st half of 2008, to $11.5 billion, according to research by PricewaterhouseCoopers [Aho09]. The primary benefits of search advertising for advertisers are its relevance and accountability. It tends to reach consumers as they enter the market for the advertised product, and advertisers ability to track consumers actions online allows for accurate measurements of advertising profitability. The downside of this accountability is a practice known as "click fraud."

### 2.3.1 Definition

Search advertisers are charged when their ads are clicked, regardless of who does the clicking. Clicks may come from potential customers, employees of rival firms, or computer programs. We refer to all clicks that do not come from potential customers as "click fraud." Click fraud is sometimes called "invalid clicks" or "unwanted clicks." This is partly because the word "fraud" has legal implications that may be difficult to prove or contrary to the interests of some of the parties involved. Google calls click fraud "invalid clicks" and says it is "any clicks or impressions that may artificially inflate an advertiser's costs or a publisher's earnings...including a publisher clicking on his own ads, a publisher encouraging clicks on his ads, automated clicking tools or traffic sources, robots, or other deceptive software." [WZ09]

The question of click fraud is vexing because search engines cannot give advertisers full information about how they detect and prevent click fraud. Doing so would be tantamount to providing unscrupulous advertisers with directions on how to commit click fraud. Advertisers are therefore forced to trust that search engines do their utmost to prevent click fraud, even though the search engines get paid every time they fail to detect

a fraudulent click. This trust was called into question in 2006 when Google CEO Eric Schmidt was quoted saying "Eventually the price that the advertiser is willing to pay for the conversion will decline because the advertiser will realize that these are bad clicks". In other words, the value of the ad declines. So, over some amount of time, the system is, in fact, self-correcting. In fact, there is a perfect economic solution, which is to let it happen." [Gho06] His remarks were interpreted as suggesting that market forces would eliminate any negative effects of click fraud in the long run, possibly undermining the need for click fraud detection. So far, there are still several negative effects of click fraud, hence the need of auditing web traffic in order to detect this type of malicious activity. [DMR+08]

### 2.3.2 Types of Click Fraud

Detailed studies were made on the types of click fraud. In 2008, the state of online fraud was summarized in [DMR+08]. It includes the representative forms of attack and respective countermeasures. The existing challenges in auditing click fraud are discussed. When done correctly, auditing helps announcers increase their trust on the market. It contextualizes the problem economically and mentions the interests involved. [WZ09] focuses mainly in how click fraud affects search engines profits in advertising. It suggests that these can benefit from this type of activity, because all clicks are paid whether they are fraud or not. It also mentions the fact that search engines cannot reveal their own methods of detecting click fraud, because these could be used to create new techniques. A description of click fraud, its background and types is given. It also suggests some general techniques and guidelines to prevent and detect click fraud. A detailed study to see if ad networks should bother fighting click fraud is made in [MWGM08]. Sometimes we tend to believe that these networks benefit from click fraud, because they get paid for every click. To validate this belief an economic market model of pay-per-click advertising is designed, focused on the effects that fraud has in this ambient. It is suggested that although "ignoring" fraud can prove to be profitable to a certain point, an ad network that actively applied fraud detection could get competitive advantage. This is because it would increase confidence in its services and hence gather more clients

#### 2.3.2.1 According to the Motivation

The two main types of click fraud, according to their motivation are:

- *Inflationary click fraud*: Search advertisements often appear on third-party websites and compensate those website owners on a per-click basis or with a share of advertising revenues. These third parties may click the ads to inflate their revenues.

- *Competitive click fraud*: Advertisers may click rival's ads with the purpose of driving up their costs or exhausting their ad budgets. When an advertiser's budget is exhausted, it exits the ad auction. A common explanation for competitive click fraud is that firms have the goal of driving up rival's advertising costs, but such an explanation may not be subgame perfect. If committing competitive click fraud is costly, then driving up competitor's costs comes at the expense of driving down one's own profits. A more convincing explanation may be found in the structure of the ad auction. When a higher-bidding advertiser exits the ad auction, its rival may claim a better ad position without paying a higher price per click.

#### 2.3.2.2 According to the Technique

The main types of click fraud, according to their technique are:

- *Manual clicking*
  - *Single Person*: Single person clicking repeatedly on the same ads, with illegal intentions or not.
  - *Click Farms*: Company dedicated to manually click on web ads 24 hours per day to either deplete a competitor's ad budget or to increase a website owner's own revenue. Some fraudsters employ teams of people in developing countries to click on ads. This may sound a little extreme but with some click bids as high as $10 - $20 each, and if you only have to pay someone $5 a day to click on links, this strategy can be very profitable for the fraudster. On May 3, 2004, the India Times published a widely read article, "India's secret army of online ad clickers." An excerpt from that article: "A growing number of housewives, college graduates, and even working professionals across metropolitan cities are rushing to click paid Internet ads to make $100 to $200 per month,"
  - *Affiliated Click Fraud*: This consists of affiliated groups of people paid to click ads. Similar to click farms, but clickers work from anywhere they want. The ad clickers receive money by clicking on website links via e-mails that the affiliating company sends, by clicking on banners and text ads in their paid-to-click section, or by referring others to the website.

- *Automated clicking*

  This subtype is where most effort is put to create sophisticated behaviors that sometimes almost replicate human behavior. Sometimes these tools have an honest purpose, such as testing web applications in terms of handling heavy load. However, one can easily see a quick way to turn these tools into click fraud "weapons".

- *Not using browsers (Browser Emulators)*

  There are some tools specially designed to emulate a web browser behavior (also known as Traffic Generators or Hit Generators), and can be used to click on specific ads in order to perform fraud. This way, defenses are eluded as it seems as legitimate clicks are made. These tools are normally coded in programming languages that efficiently manipulate text, such as Perl, or Python. Some good examples are:

  * Web Inject - This tool for automated testing of web applications and web services can be used to automatically test individual system components that have HTTP interfaces. WebInject is written in Perl and can run on any platform that a Perl interpreter can be installed on.
  * Fakezilla - Program to increase website rankings. It can generate unlimited unique fake hits anonymously to any website. Using PHP script modules, it routes unique IP addresses through thousands of proxy servers. This is a complex and sophisticated software. It can do things like generating plain hits through HTTP GET method, to submitting complex data via HTTP POST method to web forms, or simulating proxy-routed requests.
  * I-Faker - Similar to fakezilla, it sends as many unique fake hits a day as the server can handle. Using PHP, it routes HTTP get requests through a massive list of anonymous proxy servers which can be defined by the user.
  * Fake Hits Genie - Traffic generator using the same method as the above that also provides random browser and OS selection (user agent).

- *Using browsers*

  * *Using a single browser*
    - Browser plug-ins that can be used to perform click fraud: They allow anyone to click on the same paid links repeatedly from a different proxy server (thus different IP address) without switching his Internet connection (e.g. SwitchProxy for Mozilla Firefox).
    - Standalone programs to click automatically: There are a number of standalone programs that automate clicks to perform repetitive tasks. These that can be used to perform fraud on a web browser, simply by recording the coordinates and click times.
  * *Using multiple browsers*
    - On a single PC: Tools that control multiple browsers, such as Selenium. This tool works as a browser plug-in, allowing one to record a given set of actions to later reproduce them autonomously. Actually it can only record actions on Firefox, but it can launch most known

browsers executables and execute recorded actions. It uses mainly JavaScript to perform the actions, therefore being supported by most browsers. It is compatible with most OS's, and even allows the creation of code in several languages to perform the recorded actions.

- On multiple PCs: The most common type is known as Botnet - It consists of a collection of software robots, or bots, that run autonomously and automatically. The term is often associated with malicious software, because is generally used to refer to a collection of compromised computers (called zombie computers) running software, usually installed via drive-by downloads exploiting web browser vulnerabilities, worms, Trojan horses, or backdoors, under a common command-and-control infrastructure. These zombie computers can be controlled by their "bot master" to perform anything, depending of the compromised accounts/programs permissions.

- *Comparative table*

Table 2.2: Comparative table of the different techniques of click fraud.

| **Technique** | **Manual** | **Machines** | **IP Addresses** | **Geographic Origin** |
|---|---|---|---|---|
| Single Person | yes | 1 | 1 | 1 |
| Click Farm | yes | n | n | 1 |
| Affiliated Click Fraud | yes | n | n | n |
| Browser Emulators | no | 1 | n | n |
| Browser Plug-ins | no | 1 | n | n |
| Standalone Emulators | no | 1 | 1 | 1 |
| Multiple Browser Controllers | no | 1 | 1 | 1 |
| Botnets | no | n | n | n |

Table B.1 allows a fast comparison of all the described techniques. Only the most common scenarios are described. Depending on the level of sophistication of the fraudsters, there may be variations.

### 2.3.3 Existing Approaches for Click Fraud Detection

Over the last years several studies have been made, and some approaches were suggested to address this problem. In [Tuz06], Google's business model (based on advertising) is studied. The efforts made by the company to prevent/detect click fraud are also documented. A technique defined as "Badvertisments" is introduced in [GGJR06], where web masters corrupt the JavaScript code downloaded by their clients to display the ads. A part of this code goes thru the ads and performs an artificial click, increasing the web masters' revenues. Another solution to the same problem was later suggested in [PZCG10],

where a framework and several methods to detect these malicious scripts are proposed. It introduces the concept of impression-click identifier, that represents the time when the ad was legitimately requested by the page (i.e. printed). This is later used to verify if the clicks were valid or not, becoming an efective way to detect this type of fraud. Once more this is a technique to be applied on the ad syndicators servers (e.g Google), and it has proved a very low false negative and false positive rate. In 2007 a botnet ascending to 100.000 was detected by the Google team and a study was performed on its architecture in [DSTT07]. The detection methods employed at Google servers are described, and an overview to search engines general functioning is made. Other studies on Botnets can be found in [SGJ07, CL07].

Some more methods to address the problem on the content delivery systems (such as search engines) were suggested in [MKR07]. Another suggestion to fight click fraud is made in [JSJ07], where the concept of Premium Clicks is introduced. These clicks correspond to clicks of users already authenticated on other sites, or that have already bought products and therefore are considered "premium". However, this technique would have to be applied in both ad syndicators (ad servers, like Google) and announcers, making it a hard task. [KWW+08] proposed a click fraud detection system based on the fusion of data from both clients and servers, made in real time. The idea is quite simple, but the problem is gathering the data and merging it. It's proved that merging different sources of data increases the quality of the detection. The concept of click score is introduced, associated to the quality of each click. This is due to the fact that determining with certainty if the click is fraudulent or not is normally extremely difficult. Another click fraud type consisting in double clicks over the ads was detected by using jumping and sliding windows on the clickstream data [ZG08]. Two algorithms that perform a single pass over the stream data are proposed. Proof that the algorithms are efficient and effective is given after theoretical and experimental analysis.

## 2.4 Web Usage Mining

With the continuous growth of the Internet and the gigantic amounts of data generated from its usage, the interest in analyzing such data in order to extract useful information also grew, giving birth to what today is known as "Web Mining". Due to the extreme diversity of the data collected and the several applications possible for such knowledge extraction, this big field of research was then subdivided into three major research streams according to the type of data being mined, namely Web Content Mining, Web Structure Mining and Web Usage Mining. Web Content Mining focuses on extracting useful information from Web content/documents/data, Web Structure Mining tries to discover interesting patterns from the link structure of the Web and finally Web Usage Mining is specialized in extracting knowledge from user interactions with the Web. [KB00]

Web Usage Mining can then be defined as the automatic discovery and analysis of patterns in clickstream and associated data collected or generated as a result of user interactions with Web resources on one or more Web sites. It's goal is to capture, model, and analyze the behavioral patterns and profiles of users interacting with a Web site. The discovered patterns are usually represented as collections of pages, objects, or resources that are frequently accessed by groups of users with common needs or interests. [Mob07]

### 2.4.1 Data Collection and Pre-Processing

The data collection process can be seen as the retrieval of available and relevant information for the analysis. The most frequent source of data are web server logs, although other sources such as registration data, demographic data, JavaScript interaction data, etc. can be used to enrich the final data set.

Like any other data mining application the creation of a suitable data set to which algorithms can be applied fruitfully is an important task. In Web Usage Mining this is even more important due to the characteristics of the collected data. The data preparation process is often the most time consuming and computationally intensive step in the web usage mining process. It often requires the use of special algorithms and heuristics not commonly applied in other domains, and is critical to the success of extracting useful patterns from the data. This process may involve pre-processing the original data, integrating the data from multiple sources, and transforming the integrated data into a form suitable for input into data mining tasks. Usually, data preparation consists of data fusion and cleaning, page view identification, user and session identification, and path completion. [Mob07]

Data integration (or data fusion) refers normally to merging web logs with any additional information, such as registration information, user demographic information, etc. The latter are normally located in separate servers from the web log data. Data cleaning is especially needed because when loading a particular page, the browser also requests all the objects in the page (e.g. images), and each request is logged separately. Therefore, all of these log entries must be aggregated into page views at this stage. Another data cleaning task can be eliminating entries which refer to nonhuman access, such as web spiders, crawlers, etc. The behavior of such bots normally differs from human one, and isn't usually considered interesting for the analysis. Afterthe data is clean, distinct users are identified normally by combining IP addresses information with available cookie and registration data. User sessions are then detected, determining for each visit the pages that were requested, the order in which they were accessed, and the duration of each page view. An estimate of when the user left the Web site is also made, when possible. Another step is to perform path completion, because many people use the "Back" button on their browsers to return to a previously viewed page. In these situations, the browser returns to

a locally cached page instead of accessing the web server again. This forms "holes" in the web server's records of the users's path through the Web site, and information about the site topology must be used to complete these paths. [ML07] A complete discussion of the stages and tasks in the data preparation for Web usage mining can be found in [CMS99]. After this process, a series of other classic data mining pre-processing tasks is needed, in order to prepare the data to be used in the different algorithms. These tasks can be handling missing data, identifying misclassifications and outliers, and normalizing and standardizing the data

### 2.4.2 Techniques / Taxonomy

Depending on the ultimate goals of the analyst and the desired outcomes, one or more techniques can be applied to the prepared data. Here I will describe the most frequent types of pattern discovery and analysis techniques employed in Web Usage Mining problems.

#### 2.4.2.1 Exploratory Data Analysis (EDA) (Statistical Analysis) and Online Analytical Processing (OLAP)

Before we begin modeling the web usage data, it is helpful to perform some exploratory data analysis (EDA). We can learn a lot about user behavior using some simple EDA techniques as described in [Lar05]. Generally, EDA allows the analyst to see deeper into the data set, observe relationships between variables, and reveal interesting subsets of the records. To do this, the data is aggregated in units such as days, sessions, visitors, or domains. Insight knowledge about user behavior can be extracted by applying normal statistical methods to this data. The results of this analysis may be information about average view time of a page, average length of a path through a site, most frequently accessed pages, common entry and exit page, and other measures. Although this analysis may be superficial, the information revealed can be useful to direct the investigation downstream. It can also be useful for improving the system performance, and aid in marketing decisions.

Other form of analyzing this data is thru Online Analytical Processing (OLAP). The data source for OLAP analysis is normally a multidimensional data warehouse which integrates different data at different levels of aggregation, for each dimension. This provides much more flexibility, as OLAP tools allow changing aggregation levels for each dimension during the analysis. Dimensions in this specific type of data may be time duration, user agent, requested resource, and referrers. This way, the analysis can be restricted to a given time period, or at different levels of abstraction for the URL path structure. Important business intelligence metrics can also be extracted with OLAP by integrating

e-commerce data in the data warehouse. The output from OLAP processes can also be used as an input for a variety of data mining or data visualization tools.

#### 2.4.2.2 Cluster Analysis

Clustering refers to the grouping of records, observations, or cases into classes of similar objets. A cluster is a collection of records that are similar to one another and dissimilar to records in other clusters. In this specific domain, user clusters and page clusters are typically the interesting types of clusters extracted. Clustering users normally aims at finding users with similar browsing behaviors. This is done by analyzing user records in session or transaction data.

After mapping user transactions into a multi-dimensional vector space, classic algorithms such as k-means partition this space into groups of transactions that are close to each other. It does so by using a measure of distance or similarity across the vectors. The resulting clusters will reflect user groups or segments based on whatever attributes where selected to form the initial dataset. However, we must analyze each cluster in order to extract useful business intelligence, or to use it for personalized content. This is due to the fact that clusters may sometimes have thousands of data points, and therefore do not provide an aggregated view of common user patterns. To surpass this problem, one can create an aggregate view of each cluster by computing its centroid (the mean vector). Using this measure we can calculate the distance of each data point to its centroid and filter the ones that are most significant to the cluster (i.e. the ones that are closer). The resulting set of vectors can be viewed as an "aggregate user profile" accurately representing the interests or behavior of a significant group of users. This representation can be used directly to predictive modeling for automatic segmentation, and in applications such as recommender systems.

Clustering pages or items can be performed based on usage data (user sessions or transaction data), or based on the content associated with pages or items (keywords or product attributes). Content-based clustering results can be collections of pages or products related to the same topic or category. Usage-based clustering normally results in groups of items commonly searched or bought together. It can also be used to generate permanent or dynamic HTML pages suggesting related hyperlinks to users according to their past history. Some stochastic methods have also been applied to perform clustering of user transactions, normally for user modeling. Recent work shows that mixture models are able to capture more complex, dynamic behavior. This is in part due to some web applications containing very complex data, that cannot be modeled by basic probability distributions. For instance, each user may have different types of behavior according to the task, and each behavior can be modeled by a different distribution.

Mixture models (such as mixture of Markov models) assume that there exist k types of user behavior (or k user clusters) in the data. Also, each user session is assumed to be generated via a process that models the probability distribution of the observed variables and hidden variables. It starts by choosing a user cluster with some probability. Then, the user session is generated from a Markov model with parameters specific to that user cluster. Both the probability of each cluster and the parameters of each mixture component are normally estimated using the Expectation-Maximization (EM) algorithm [DDLR77]. Mixture based user models can be very flexible. For instance, a mixture of first-order Markov models [CHM+03] can not only probabilistically cluster user sessions based on similarities in navigation behavior, but also characterize each type of user behavior using a first-order Markov model. This way it can also capture popular navigation paths or characteristics of each user cluster. A mixture of hidden Markov models has also been proposed [YH03] for modeling clickstream of web surfers. This approach can also be used for automatic page classification. However, mixture models have their own deficiencies. As each individual observation (such as a user session) is generated from only one component model, the probability assignment to each component only measures the uncertainty about this assignment. This assumption limits this model's ability of capturing complex user behavior, and may even result in overfitting. Probabilistic Latent semantic Analysis (PLSA) provides a reasonable solution to the mentioned problem [Hof01]. In a web user navigation scenario, each observation (user visiting a page) is assumed to be generated based on hidden (unobserved) variables that "explain" the user-page observations. The data generation works like this: a user is selected with a certain probability, and conditioned on the user a hidden variable is selected. Then, the page to visit is selected conditioned on the chosen hidden variable. Since each user usually visits multiple pages, this data generation process certifies that each user is explicitly associated with multiple hidden variables, thus reducing the overfitting problem described above. The PLSA model also uses the EM algorithm to estimate the parameters which probabilistically characterize the hidden variables, and to measure the relationship among hidden and observed variables.

Again, one of the main advantages of PLSA in web usage mining is that using probabilistic inference with the above estimated parameters, we can derive relationships among users, among pages, and between users and pages. Concluding, this framework provides a flexible approach to model a variety of types of usage patterns.

#### 2.4.2.3 Association and Correlation Analysis

Association rule discovery and statistical correlation analysis can find groups of items or pages that are commonly accessed or purchased together. This enables Web sites to organize the content better, or to provide effective cross-sale product recommendations.

Most common approaches to association discovery are based on the Apriori algorithm. This algorithm finds groups of items (page-views) occurring frequently together in many transactions (i.e., satisfying a user specified minimum support threshold). These groups of items are referred to as frequent itemsets. Association rules which satisfy a minimum confidence threshold are then generated from these frequent itemsets. An association rule is an expression of the form $X \rightarrow Y$ [sup, conf], where X and Y are itemsets, sup is the support of the itemset $X \cup Y$ representing the probability that X and Y occur together in a transaction, and conf is the confidence of the rule, defined by $sup(X \cup Y)/sup(X)$, representing the conditional probability that Y occurs in a transaction given that X has occurred in that transaction.

The mining of association rules in Web transaction data has many advantages. For example, a high-confidence rule such as special-offers/, /products/software/ $\rightarrow$ shopping-cart/ might provide some indication that a promotional campaign on software products is positively affecting online sales. Such rules can also be used to optimize the structure of the site. For instance, if a site does not have a direct link between pages A and B, the discovery of a rule, $A \rightarrow B$, would indicates that providing a direct hyperlink from A to B might aid users in finding the intended information. However, association rule recommendation systems have a problem. They cannot give any recommendations when any given user visits (or rates) only a very small fraction of the available items. Therefore it is often difficult to find a sufficient number of common items in multiple user profiles. Dimensionality reduction was proposed [FBH00] to tackle this problem, but it results in possible loss of interesting items. Other potential solution was to rank the discovered rules by the degree of intersection between the left-hand side of the rule and the user's active session. Then the top k recommendations would be shown. This avoids having to find an exact match with the left-hand-side of the rules. Yet another solution was to use collaborative filtering, that explores the concept of neighbor to show recommendations to a given user based on the closest neighbors histories.

Because it is difficult to find matching rule antecedent with a full user profile (e.g., a full user session or transaction), association-based recommendation algorithms typically use a sliding window w over the target user's active profile or session. The window represents the portion of user's history that will be used to predict future user actions (based on matches with the left-hand sides of the discovered rules). The size of this window is iteratively decreased until an exact match with the antecedent of a rule is found. The problem with a naive approach to this algorithm is that it requires repeated search through the rule-base. However, efficient trie-based data structures are used to store the discovered itemsets and allow for efficient generation of recommendations without the need to generate all association rules from frequent itemsets [MDLN01]. In these structures the frequent itemsets are stored in a directed acyclic graph. The graph is organized into levels from 0 to k, where k is the maximum size among all frequent itemsets. Each node at

depth d in the graph corresponds to an itemset, X, of size d and is linked to itemsets of size d+1 that contain X at level d+1. The single root node at level 0 corresponds to the empty itemset. To be able to search for different orderings of an itemset, all itemsets are sorted in lexicographic order before being inserted into the graph.

A recommendation engine based on this framework matches the current user session window with the previously discovered frequent itemsets to find candidate items (pages) for recommendation. Given an active session window w and a group of frequent itemsets, the algorithm considers all the frequent itemsets of size |w| + 1 containing the current session window by performing a depth-first search of the Frequent Itemset Graph to level |w|. The recommendation value of each candidate is based on the confidence of the corresponding association rule whose consequent is the singleton containing the page to be recommended. If a match is found, then the children of the matching node n containing w are used to generate candidate recommendations. In practice, the window w can be incrementally decreased until a match is found with and itemset. A problem with using a single global minimum support threshold in association rule mining is that the discovered patterns will not include "rare" but important items which may not occur frequently in the transaction data. This is particularly important when dealing with Web usage data. It is often that references to deeper content or product-oriented pages occur far less frequently than those of top level navigation-oriented pages.

Yet, for effective Web personalization, it is important to capture patterns and generate recommendations that contain these items. A mining method based on multiple minimum supports was proposed in [LHM99] that allows users to specify different support values for different items. In this method, the support of an itemset is defined as the minimum support of all items contained in the itemset. The specification of multiple minimum supports thus allows frequent itemsets to potentially contain rare items which are deemed important. It has been shown that the use of multiple support association rules in the context of Web personalization can dramatically increase the coverage (or recall) of recommendations while maintaining a reasonable precision [MDLN01].

#### 2.4.2.4 Sequential and Navigational Patterns Analysis

Sequential pattern mining techniques aim at discovering inter-session patterns, as the presence of a set of items followed by another item in a time-series session data. These patterns can then be used by marketeers to predict future visit patterns which can be helpful in placing targeted advertisements to certain users. Other forms of temporal analysis that can be made on these patterns include trend analysis, change point detection, or similarity analysis. In the context of web usage data, these techniques are employed to capture frequent navigational paths among user trails.

Sequential patterns (SPs) in web usage data capture the paths of pages often visited by users, in the order they were visited. Sequential patterns are those sequences of items that frequently occur in a sufficiently large proportion of transactions. A sequential pattern is considered contiguous if each pair of adjacent items appear consecutively in a transaction which supports the pattern. A normal sequential pattern can represent non-contiguous frequent sequences in the underlying set of sequence transactions.

Given a sequence transaction set T, the support (denoted by sup(S)) of a sequential (respectively, contiguous sequential) pattern S in T is the fraction of transactions in T that contain S. The confidence of the rule $X \rightarrow Y$, where X and Y are (contiguous) sequential patterns, is defined as: $conf(X \rightarrow Y) = sup(X \circ Y)/sup(X)$, where $\circ$ denotes the concatenation operator.

In the context of Web usage data, contiguous sequential patterns (CSPs) can be used to capture frequent navigational paths among user trails [SF99]. In contrast, items appearing in SPs, while preserving the underlying ordering, need not be adjacent, and thus they represent more general navigational patterns within the site.

Viewing web transactions as sequences of pageviews allows using useful and well-studied models to be applied in finding or discovering user navigation patterns. One of these approaches is to model navigational activities in a web site as a Markov model. Here, each pageview can be represented as as state and the transition probability between states can represent the likelihood that a user will navigate from one state to the other. This representation facilitates the calculation of useful site or user metrics. Using these structures one can compute the probability that a user will make a purchase, given that she performed a search in an online catalog. Markov models have been proposed as a tool to perform link prediction and web pre-fetching to minimize latencies (system optimization) [DK04, Sar00]. The goal in these applications is to make predictions on the next user action, based on previous behavior. They have also been applied to discover high probability user navigational paths in a web site [BL00].

Basically, a Markov model is characterized by a set of states $s_1, s_2, ..., s_n$ and a transition probability matrix, $[Pr_{i,j}]_{n \times n}, where Pr_{i,j}$ represents the probability of a transition from state $s_i$ to state $s_j$. Each state represents a contiguous subsequence of prior events. The order of the Markov model corresponds to the number of prior events used in predicting a future event. this way, a kth-order Markov model predicts the probability of the next event looking at the past k events. To calculate the probability of reaching a state sj from state si via a non-cyclic path, we simply have to multiply all the transition probabilities along the path.

Higher order Markov models normally provide a higher prediction accuracy. However, this usually results in lower recall (or coverage), and much higher complexity of the model due to the larger number of states. All-kth-order Markov models for coverage improvement reduction, and a new state reduction technique that reduces model size,

called longest repeating subsequences (LRS) were proposed in [PP99]. These all-kth-order Markov models may require the generation of separate models, as if a kth order model cannot make the prediction, it will attempt to predict by incrementally decreasing the order. This can lead to even higher space complexity because it has to represent all possible states for each k. In [DK04] selective Markov models were proposed to tackle these complexity problems. The proposed schemes involve pruning the model based on criteria such as support, confidence and error rate.

Another approach to efficiently representing contiguous navigational paths is by inserting each path into a trie structure. A good example of this is the notion of aggregate tree introduced as part of the WUM (Web Utilization Miner) system [SF99]. This tree is built by transforming the transactions from web logs into sequences, and merging the sequences with the same prefix into the aggregate tree (a trie structure). This way each node represents a navigational subsequence from the root (empty node) to a page and also contains the frequency of occurrences of the subsequence. WUM uses a mining query language called MINT to discover generalized navigational patterns from this trie structure. MINT also includes mechanisms to specify sophisticated constraints on pattern templates, such as wildcards with user-specified boundaries, as well as other statistical thresholds such as support and confidence.

#### 2.4.2.5 Classification and Prediction

Classification consists in mapping a data item into one of several predefined classes. In this particular domain, we are interested in developing profiles of users belonging to a particular category (or class). This requires extraction and selection of features that best describe the properties of a given class. Supervised classification can be done using algorithms such as decision trees, naive Bayesian classifiers, k-nearest neighbor classifiers, and Support Vector Machines (SVM). One typical approach is to use discovered clusters and association rules for training these classifiers, using the outputs of the first ones as the classes for the latter. Classification techniques play an important part in web analytics applications for modeling the users according to various predefined metrics. For example, a classification model can be built to classify users according to their propensity to buy or not. This model could be created using the sums of bought items by users during a period of time together with demographic and navigational pattern data. Another important application of classification and prediction in the web domain is collaborative filtering. Most collaborative filtering applications in existing recommender systems use k-nearest neighbor classifiers to predict user ratings or purchase propensity. This is done by measuring the correlations between a target user's profile (which may be a set of item ratings or a set of visited or purchased items) and past user profiles in order to find users with similar characteristics or preferences [HKTR04]. Basically, collaborative filtering based

on k-nearest neighbor (k-NN) functions by comparing the activity record of a target user with the historical records of other users, choosing the top k users that have similar tastes or interests. The mapping of a user record to its neighborhood can be done using similarity in rating of items, accesses to similar content or pages, or purchase of similar items. Normally, in these type of applications the user records are a set of rating for a subset of products. The identified neighborhood is then used to recommend items not already purchased or visited by the active user. this way, we can identify two main phases in collaborative filtering: neighborhood formation and recommendation phase. The problem in user-based formulation of the collaborative filtering approach is the lack of scalability, because it requires real-time comparison of the target user to all user records in order to generate predictions. A solution to this problem, named item-based collaborative filtering was proposed in [SKKR01]. This approach works by comparing items based on their pattern of ratings across users. Once more, k-NN can be used, attempting to find the k similar items that are co-rated by different users similarly.

### 2.4.3 Applications

Web Usage Mining has been applied mainly with two big objectives in mind, either to build better sites or to know visitors better (business intelligence). These main goals can result in several specific applications:

#### 2.4.3.1 Bulding better sites

- **System Improvement**

  One of the key factors for a user to keep using a given service is its performance and reliability. The same criterion applies to any web service, and because of this constant efforts are made to improve the quality of these systems. Web Usage Mining can be applied in order to achieve this, by discovering frequently accessed pages to perform caching, detecting server load time periods to balance load and to know when a given system should become distributed to preserve service quality [CKR98]. Another big application is related to security, where these techniques are used to detect intruders, fraud, attempted break-ins, etc. Because of the advent of dynamic content, the benefits of caching were reduced at both the client and server sides, and efforts are being made to detect path profiles that are then used to generate dynamic pages based on the current profile, reducing page generation latency [SKS98].

- **Site Modification**

  As we all know users are attracted to websites that please them whether visually, organizationally or in terms of content. Given this fact we may conclude that these

factors are crucial to the success of an application, and Web Usage Mining provides the feedback necessary to understand user behavior on our website, giving the web designer precious information on how to redesign the website. This usage data can be considered as a website's constant usability test, although the information is not as complete as a formal one. Some typical measures can be page access time, page abandonment rate or page popularity [GKM99], and page clustering is often used to determine which pages should be directly linked [SN03]. User navigational patterns can also be used to modify the structure of a website [BS00], by shortening the length of the paths for the most visited pages for instance.

- **Search Engine Optimization**

  For today's majority of Internet users, their window to the Web are search engines. It's of interest to search engines to know how people search, that links do they click and in what order, and how do they refine their search, among many others. The work done by [ZD02] explores relationships among users, queries and resources to improve a search engine. Search engine optimization can also refer to optimizing a website's content to appear on a better position on search results. As we well know, people tend to click on the first links, or the higher ranked, because search engines found them to be the more relevant to the search query. Web Usage Mining can be used to optimize the tags for search engine indexing, optimize its content for search engines web crawlers, etc.

#### 2.4.3.2 Business Intelligence

- **Customer Behaviour**

  Customer behavior information is of priceless value for marketeers and as expected, this kind of information has the same value on e-commerce websites. Thru Web Usage Mining we are able to obtain these behavior patterns that aid marketeers in developing better campaigns, creating promotions and predicting customer behavior itself. Valuable intelligence on the customer relationship cycles can be extracted, specifically customer attraction, retention, cross sales and customer departure [BM98].

- **Personalization / Recommender Systems**

  Many websites would like to give its users a personalized experience, especially e-commerce applications. In this specific case, personalization is mostly used under the form of recommendations to the user, according to her/his profile and behavior [MCS00, MDLN01, MDLN02, JZM05, Mob07]. This is very attractive to these applications, because it favors for instance cross-sales and up-sales. Web Usage

Mining is used to find products that are normally bought together, or that a specific user profile normally buys/has bought a set of products [CKK02]. This can also be applied to pages viewed inside the website, recommending links. Clustering can be used to perform market segmentation in e-commerce applications or provide personalized web content (e.g. recommendations [MDLN02]) to groups of users sharing the same interests [PPKS02]. Analyzing deeper these user clusters using demographic attributes can result in the discovery of valuable business intelligence. Association analysis (among products or pageviews) and statistical correlation analysis (generally among customers or visitors) have been used successfully in Web personalization and recommender systems [HKTR04, MDLN01]. Many techniques discussed earlier can be used to automatically discover user models and then apply them to provide personalized content to an active user [JZM05, PPPS03].

- **Usage Characterization**

  Characterizing how users interact with a specific website or even with the internet can provide valuable insight in the design of better websites and web browsers. This is done by capturing the interface usage and the navigational strategies, which can then be analyzed in order to make such improvements [Spi00]. The WUM (Web Utilization Miner) system proposed here, which performed navigational pattern analysis, has been proved successful in evaluating also the navigational design of a web site.

### 2.4.4 Comparative Table

Table 2.3: Comparative table of the different techniques of Web usage mining

| **Technique** | **ST** | **GEO** | **NU** | **NO** | **SI** | **SM** | **SEO** | **CB** | **PR** | **UC** |
|---|---|---|---|---|---|---|---|---|---|---|
| EDA / OLAP | X | | X | X | X | | X | X | | |
| Clustering | X | X | X | X | X | | X | X | X | |
| Association Rules | X | | | X | X | X | X | X | X | |
| Sequential Patterns | X | | | X | X | X | | | X | |
| Classification | X | X | X | X | X | X | | X | X | X |

Legend:

ST: Statistical Methods
GEO: Geometrical Methods

NU: Numerical Values
NO: Nominal Values

SI: System Improvement
SM: Site Modification
SEO: Search Engine Optimization
CB: Customer Behavior
PR: Personalization
UC: Usage Characterization

Table B.1 enables a fast comparison of all the described techniques. Although some techniques are more applied to some specific endings, we can see that the different techniques can be applied in a vast number of applications. This versatility depends on how the data is manipulated in order to achieve the desired objective. Another observation one may extract is that all methods can handle nominal attributes (i.e. textual). Numerical attributes can only be handled without any manipulation by clustering and classification.

## 2.5 Conclusions

From the analysis of the related work, several conclusions could be made. These conclusions defined essentially the methodological approach.

The AuditMark's AuditService analysis briefly described the auditing platform where the solution will be integrated. The architecture and the most relevant modules were described. Special attention was given to the Tag-Based Data Collection module, and more specifically to the JavaScript Interaction Code (JIC). Some of the most important attributes collected by the JIC were mentioned, but the complete list was not mentioned for compromising proprietary information. It can be said however that including the data collected thru Java, the total amount of attributes is close to 300 (although there is some attribute redundancy).

Even so, it is clear that the gathered data is extremely rich. It contains all the attributes traditional web server logs provide, as well as many more attributes that would be impossible to extract from traditional web server logs. However, there is one serious limitation, which is the fact that the retrieved information is only relative to the landing page. This seriously limits navigational pattern analysis. By knowing the pages the user visited, one could extract valuable information such as paths and page visit times.

In a Fraud Detection problem, the appropriate method to use depends on the available data's characteristics. If labelled cases of both fraudulent and legitimate cases exist, one can consider using supervised methods. Otherwise, only unsupervised methods can be employed. Considering the problem at hands, where there are no known examples of both types, the usage of unsupervised methods became clear. A review of the used metrics was also presented, with special emphasis to the classical measures resulting from the binary classifier confusion matrix. It was observed that several approaches use suspicion scores as the detection output as well. From a validation point of view, the quality of these scores can be seen as a performance indicator. The set of metrics used in this approach were adapted from Click Fraud detection scenarios.

After analyzing the Click Fraud problem and existing approaches, one can conclude that this problem cannot be completely solved by a single approach. This is mainly due to the vast amount of existing techniques, each with its specific variations. Thus, creating a method that accurately detects all of them is virtually impossible. Fraudsters are

constantly creating new click fraud techniques. Therefore every approach must be able to adapt and will be in constant improvement. Another conclusion is that approaches should be designed for a specific point of the business. This increases the probability of the approach being effectively deployed, since fewer entities are affected. In the proposed approach, only a specific type of click fraud is being targeted. The type of fraud being detected consists of those attacks that are likely to cause deviations in normal traffic patterns (such as click farms, affiliated networks and botnets). Finally, the solution will be implemented only in the advertiser's web sites. Even though less information is available, this makes the deployment easier to accomplish. The following section will present a complete review of Web Usage Mining state-of-the-art. It will allow a precise definition of the most adequate techniques to employ in the context of this click fraud detection.

The Web Usage Mining techniques that can be employed on a given problem also depend primarily on the available data's characteristics. The decision of using unsupervised techniques had already been made when studying Fraud Detection's state-of-the-art. Hence, after analyzing the available data and the problem at hands, the chosen techniques were EDA and cluster analysis. EDA provides extremely valuable information on the data. In this context, it models the base traffic distributions and detects deviations from it. These deviations allow differentiating some clicks that are suspect of being invalid. To differentiate these groups of clicks that are potentially invalid, the choice of using cluster analysis became clear. Additionally, given the nature of the click fraud problem, marking a given click as invalid is very context-dependant, and extremely difficult. Hence, no labels (valid or invalid) can be given to the clicks. This leads to unlabelled datasets, which automatically exclude supervised techniques such as classification and regression. As a consequence of the data collection method being used, only the clicks performed on the landing pages (pages where the ads point to) had their information collected. Thus, no sessions could be extracted because the collected clicks all belonged to the same page. So, techniques that explore user session data were automatically discarded. These are Sequential and Navigational patterns analysis techniques. In this context, Association and Correlation analysis techniques have the potential of describing the traffic's characteristics in terms of rules. This could aid in characterizing the traffic patterns, to then detect deviations. Even though, they were not applied and were left for future developments.

The used methodological approach was defined as a result of studying all the scientific areas related to the dissertation theme. This methodology is described in detail in Chapter 3.

# Chapter 3

# Methodology

## 3.1 Introduction

AuditMark is a company that is able to analyze its customers' (online advertisers) web traffic by using the AuditService auditing platform. To do so, a JavaScript call to the AuditService data collection module is inserted into the customer's website code. This call is known as Javascript Interaction Code (JIC) and is inserted on the landing pages (pages to where the ads point to), and triggers the data collection process. Then, for each click made on the customer ads, the information about the visitor is gathered, to be analyzed later. This information allows the characterization of the visitor who performed each click (e.g. operative system, Web browser, geographic location, user language and screen resolution). The analysis of this information results in the attribution of a score to each click, which is then used to measure the analyzed website's traffic quality.

So far, the clicks are analyzed individually and the context relative to the underlying traffic patterns is ignored. This thesis proposes to incorporate the usage of Web Usage Mining techniques into the AuditService platform. By analyzing the traffic patterns of the website, these techniques allow the detection of deviations from the normal traffic patterns. These deviations consist of sets of clicks, that when analyzed individually would be completely valid. However, when analyzed considering the website's normal traffic patterns they reflect a deviation, which can be considered a potential organized attack. These organized types of click fraud are not detectable using the current techniques used by the AuditService.

In this approach, the use of synthetic data was required for the validation process. In a click fraud detection scenario, proving the validity of a given click depends mostly on the context and is sometimes impossible. When analyzing real web traffic data, there is never absolute certainty that a click is valid or invalid. Since fraudsters cannot be "hired" to commit fraud on a given web site and then report back, the uncertainty about the validity of the clicks remains. Using synthetic data it is possible to compare the result

of the analysis with the correct answer known apriori to be correct. Therefore, to evaluate the performance of the proposed solution, a generated dataset was used as input for the analysis. The generated dataset includes both valid and invalid traffic, where the former was created to mimic the behavior of common click fraud techniques. These include single person attacks, click farms, affiliated networks, automated-click tools and botnets.

The generated dataset was created using a traffic simulator developed for this purpose. In each run, it generates a set of clicks between two dates using stochastic methods, and outputs a simulation log file. This log file is then used to perform the clicks, simulating a real scenario. A web page is designed to emulate several ads, which lead to the landing page containing the call to the AuditService data collection module. The data relative to each click is then collected by the AuditService collection module and saved for posterior analysis. This data contains some additional information about the type of click (valid or invalid), for validation purposes. Some preprocessing is made on this generated dataset, which leaves it ready for the first step of the analysis.

The data between the two dates used for the traffic generation constitutes then the analysis time period for the remaining process. This time period is then split into several time windows, which are then overlapped. By overlapping the windows it becomes possible to analyze the traffic patterns that repeat over time. These time windows are split once more into smaller time slices called time units, to enable the comparison between equivalent periods across all time windows. Grouping together the same time unit from all time windows forms the analysis subsets. Consider that the analysis time period was one month (e.g 30 days), the time window was set to one day, and the time unit was set to one hour. This analysis time period data would be divided into 30 different time windows (days), and then each time unit would be divided into 24 time units (hours). An example of an analysis subset could be the 30 time units corresponding to the 13:00 to 14:00 (1 hour), from all the time windows (days). One can see that each of these analysis subsets will contain the traffic pattern during each hour (time unit), for every day (time window) of the month (analysis period).

Once the analysis period is divided into these analysis subsets, a statistical analysis models the base distribution (normal traffic pattern) and then detects deviations (outliers) from this distribution. Given that the base distribution is not known apriori, it must be modeled using the data itself. Basing the statistical analysis on the central limit theorem, the base distribution of each analysis subset is considered to be approximately normal. This enabled the usage of a statistical test designed for normal data, the modified z-score test. In a modified z-score test the z-score is determined based on outlier resistant estimators. The median and the median of absolute deviation about the median (MAD) are such estimators. Using these estimators, the median is considered to be the mean value that characterizes the base distribution. The modified z-score is then calculated for each element of the analysis subset. An element is labeled as an outliers when it has a z-score

greater than a defined threshold. This outlier detection method is applied in several dimensions of the data: absolute number of clicks, number of clicks per os, number of clicks per web browser, number of clicks per country and number of clicks per referrer. The results are deviations from normal traffic patterns, each one in a specific point in time of the analysis period, with specific characteristics. These deviations are now considered suspicious attacks of click fraud.

After the deviations have been detected in all time units and all dimensions, they are analyzed to characterize the suspicious attacks. This analysis groups dimensions that have a similar amount of deviation when compared to the base distribution. By doing so, only those deviations that really characterize the possible attack are grouped. The result of this attack characterization is a set of pairs $attribute = value$, where the attribute is the dimension, and the value the category of the dimension responsible for the deviation. It also contains an estimate of the amount of clicks that caused the deviation (possible attack size). Once all deviations are analyzed and the possible attacks characterized, this information is used to try and separate the clicks that caused the deviation from those that belong to the base distribution.

To differentiate the clicks that caused the deviation from those that belong to the distribution, a clustering algorithm is used. First, the complete data corresponding to each deviation time period (the time unit of the time window) is retrieved from the database and preprocessed. Each deviation data forms a dataset. Then, using each deviation analysis result, the clustering algorithm is configured accordingly and is applied to the respective dataset. From the resulting cluster set, a cluster is selected if it is sufficiently similar to the suspicious attack characteristics. The choice is made comparing each cluster's size with the estimated deviation amount, and the cluster's attribute values with the attack characteristics (pairs $attribute = value$).

For each cluster that is selected as containing a click fraud attack, a suspicion score is given to all its elements. This score can me seen as a measure of traffic quality expressed in percentage, where 100% corresponds to a completely valid click, and 0% to an undeniable case of click fraud. The current score formula is quite simple and is calculated based on the similarity between the selected cluster size and the estimated attack size. If the cluster size is less or equal than the estimate size it is considered to contain the entire attack and a score of 0% is given. On the contrary, if the cluster size is greater than the estimate it is considered to contain false positives (valid clicks detected as invalid) and the score is computed as score = estimated size / cluster size. For instance, if the estimated attack size was 50 and the cluster had 100 elements, the resulting score would be 50%. Thus, this reflects the "certainty" that the solution has that those clicks are in fact invalid. Even though this scoring formula is extremely simple, it proved to be quite robust during the experiments as can be seen in Chapter 4.

When the detection process ends, the result is a set of clicks marked as being suspect of

fraud, each one with a suspicion score. The final step of the methodology lies in validating these results. As mentioned above, the type (valid or invalid) of each click is already contained in the data. Analyzing this set of clicks, one can know the exact amount of: invalid clicks correctly marked as invalid (true positives); valid clicks incorrectly marked as invalid (false positives); valid clicks correctly marked as valid (true negatives) and invalid clicks correctly marked as valid (false negatives).

These amounts enable the calculation of performance metrics and the validation of the detection process. Considering that in a real life scenario one wouldn't have the information about the type of each click, the suspicion scores are also used as a performance metric. Since the suspicion scores are only altered for the clicks marked as being invalid, the average scores for both true positives and false positives are computed and evaluated. All the clicks that are not marked have 100% scores, and would not bring any information for this analysis.

This approach was implemented as a functional prototype for an automated analysis software tool, to be integrated in the AuditService afterwards. This software tool will be added to the existing family of data processing plugins of the platform, where each one explores a different analysis approach, is fully independent and contributes to the final traffic quality score. The AuditService will benefit from the inclusion of this approach, and will be able to detect some types of click fraud undetectable so far.

This chapter provides an overview of the proposed methodology in Section 3.2, and describes the traffic simulator used for the dataset generation in Section 3.3. The preprocessing steps performed on the data are detailed in Section 3.6, followed by the description of the statistical analysis in Section 3.4. The process of attack characterization is explained in Section 3.5, and Section 3.7 describes the clustering method used in the approach. Section **??** describes the click scoring process, and Section 3.9 defines the validation methodology used to test the approach. Section 3.10 summarizes some implementation details and the solution's architecture. Some approach limitations are presented in Section 3.11 and concluding remarks can be found in Section 3.12.

## 3.2 Overview of the Proposed Methodology

The proposed methodology consists of seven main steps. Some of these steps are subdivided into more than one phase. These main steps and their respective phases can be summarized as follows:

1. *Dataset generation*: Since the usage of synthetic data was necessary for validating this approach, a traffic simulator had to be developed for this purpose. This step consists of the web traffic generation procedure, which outputs the generated dataset

used for the analysis. The following phase comprises some preliminary preprocessing steps that are necessary to prepare the data for the remaining process.

2. *Statistical analysis*: This step of the process is similar to the fraud detection process commonly used when no examples of fraud are known. In these cases the common practice is modelling the base distribution to then detect deviations (outliers) from it. The same principle was employed in this approach. The phases contained in this step are modelling the base distribution and detecting the outliers.

3. *Outlier analysis*: Once the outliers are detected, they are analyzed to characterize possible click fraud attacks. The analysis of the outlier data produces attack characterizations, which are used in the next steps.

4. *Preprocessing*: In order to apply unsupervised clustering techniques, the data must first be preprocessed for optimal results. The data for each possible attack forms a dataset. Each dataset is then submitted to a set of preprocessing operations before being clustered.

5. *Clustering analysis*: Once the data of each attack is gathered and preprocessed, a clustering algorithm attempts to differentiate the clicks that effectively caused the deviation from the ones belonging to the underlying traffic pattern. For each outlier dataset, the clustering algorithm is configured using the outlier analysis performed previously. From each resulting cluster set, a cluster may or not be chosen according to the degree of similarity it has to the estimated attack characteristics.

6. *Scoring*: For each cluster that is chosen as suspect of containing a click fraud attack, a score is computed to all its elements. This score is calculated according to a very simple formula, based on the similarity the cluster size has to the estimated click fraud attack size (number of clicks).

7. *Validation*: When the entire detection process is finished, the detection quality is evaluated. This evaluation is made using metrics commonly used in other fraud detection problems. Since the generated dataset contains labels indicating the type of each click, one can calculate these performance metrics easily. The scores obtained in the last step are also used to compute two additional metrics for robustness evaluation.

Figure 1 illustrates all the main steps of the proposed methodology workflow.

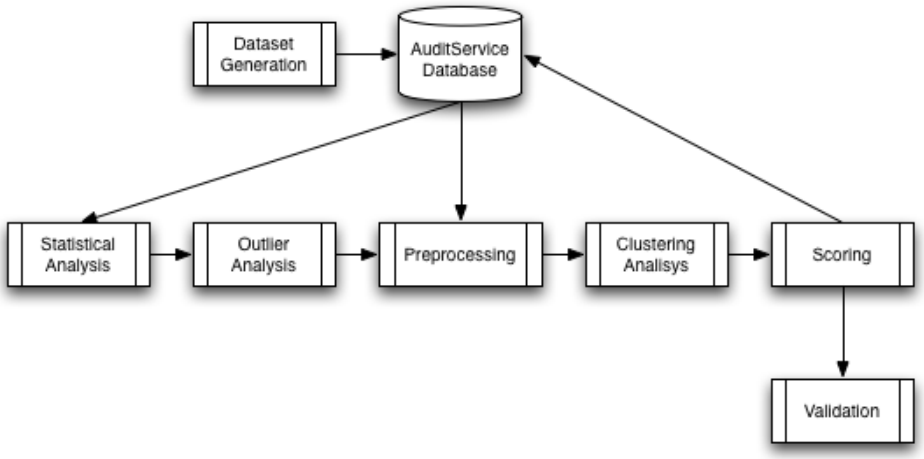


Figure 3.1: Proposed Methodology workflow.

## 3.3 Dataset Generation (Traffic Simulation)

To enable the process validation, it was necessary the usage of synthetic data. In a click fraud detection scenario, proving the validity of a given click depends mostly on the context and is sometimes impossible. When analyzing real web traffic data, there is never absolute certainty that a click is valid or invalid. And "hiring" fraudsters to commit fraud on a given website and then report back is, obviously, out of question. Thus, the uncertainty about the validity of the clicks remains. Using synthetic data allows the comparison of the analysis results with the correct answer known apriori to be correct. Therefore, to evaluate the performance of the proposed solution, a generated dataset was used as input for the analysis. The generated dataset contained both valid and invalid traffic, designed to replicate a real scenario. This dataset was created using a traffic simulator, and both will be detailed in this section.

Web traffic data can be seen as a time series dataset. So, the first step of the traffic generation is to define a start and end date for the simulation. Then, an event generator is necessary to generate the clicks. There are two well-known probability distributions to model events occurring in a fixed amount of time. These are the Poisson and the Negative Exponential distributions. The first models the number of events occurring in a given fixed time period if these events occur independently and with a known average rate. The latter describes the times between events in a Poisson process. They are equivalent and thus have the same single parameter, the average rate of the events. The only difference lies on the modelled variable (number of events vs. time between events, respectively). Both can be used to generate events, the Poisson distribution was chosen. The reason was that the used Poisson distribution generator's implementation (using the Patchwork Rejection/Inversion method) was roughly two times faster than the Negative Exponential

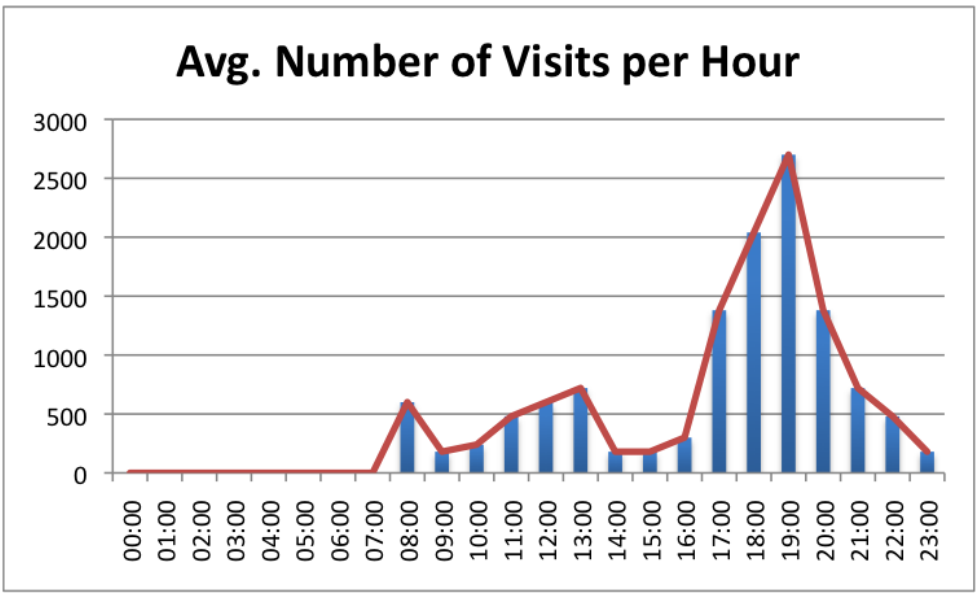


Figure 3.2: Average number of visits per hour used in the dataset generation.

distribution generator's implementation. After the event generator was chosen, the event rate had to be defined.

Real traffic doesn't follow a constant rate. Instead, it varies according to the hour of the day, day of the week, external events (e.g. sports website after a football match), etc. External factors cannot be modelled since they depend on unknown variables and are somewhat chaotic. Thus, only the average visit rates can be modelled for each specific time unit. These average visit rates can be defined for each hour of the day, each day of the week, to then generate the events accordingly. The chosen time unit is completely arbitrary, and one can use the desired time granularity. One can even create more complex structures. For instance, the average rate from 10:00 to 11:00 at Mondays may be different from the average rate for the same time interval, at Saturdays. However, the generated dataset was created using different average visit rates for each hour of the day. These average visit rates create the traffic pattern for the number of visits of the web site. After the average visit rates are defined for each time unit, the events are generated accordingly. Figure 3.2 illustrates the used average visit rates for simulating the used generated dataset.

The main challenge of the traffic simulator was being able to mimic both valid and invalid traffic, the best way possible. The first step of this process begins by defining the portion of the traffic that will be invalid. According to [Cli09] reports, it is estimated that around 14-17% of all clicks are invalid. Figure 3.3 illustrates these estimates.

[Anc10] even states that 29.2% of all clicks in Q1 2010 where attempts of click fraud, although it seems inflated. Thus, a value for this portion of invalid traffic had to be estimated. This portion can then be seen as the probability a given click has of being invalid. For the used dataset, the portion of invalid clicks was set to 15% because it

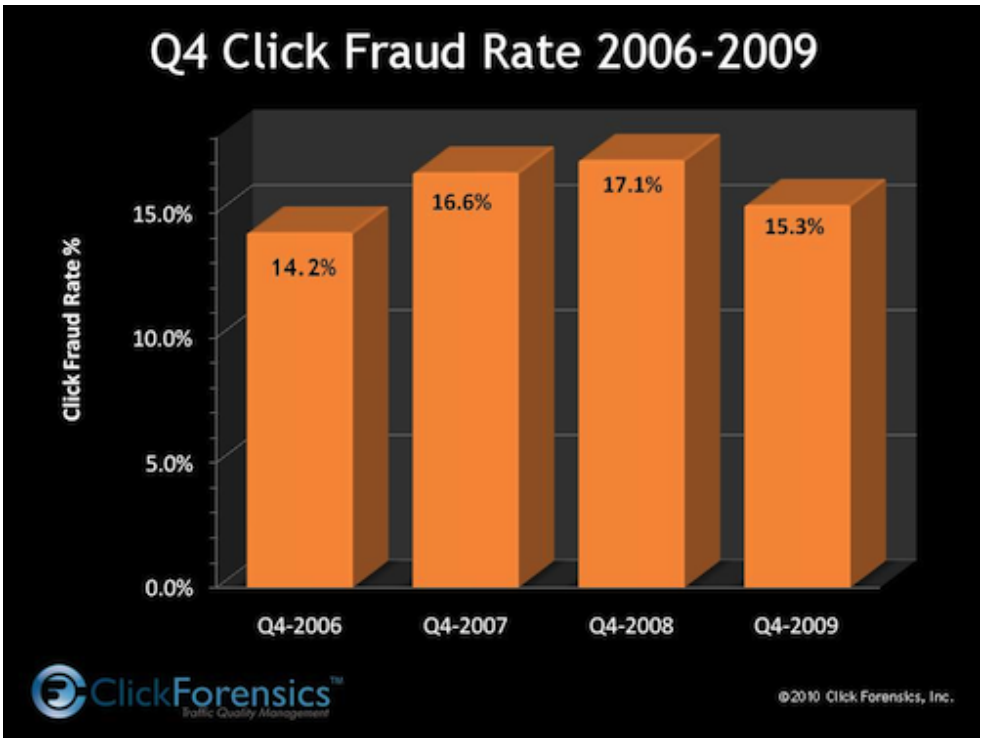


Figure 3.3: $4^{th}$ Trimester Click Fraud Rate 2006-2009 evolution.

seemed reasonable. So, an invalid click was generated with 15% probability. Using this simple method, each generated click is labelled as either being valid or invalid. Therefore, the result consists of two sets of clicks: a valid click set, and an invalid click set. These two sets of clicks are then treated separately to replicate each type accordingly.

### 3.3.1 Valid Traffic Generation

When a click is marked as being valid, its characteristics are defined using a set of stochastic operations. Probability distributions are defined for each simulated click characteristic. Currently, the probability distributions are defined for the operative system (OS), Web browser, geographic location and used ad link (referrer). For each of these click's characteristics, the value is randomly chosen according to each characteristic probability distribution. Even so, the process can be different depending on the characteristic being generated. An example is that the Web browser depends on the used OS. Hence, the probability distribution of the Web browsers was defined per OS. So, after the OS is defined, the Web browser is chosen accordingly. Another example is that the geographic location is also used to generate an IP address of that country. Using a list of the IP ranges of each country, a single range is chosen randomly. Then, an IP address belonging to that range is randomly generated. Using random events, the clicks are characterized according to these probability distributions.

In the used dataset, the probability distributions for the OS's and respective Web browsers were based on usage statistics available online at [w3s10]. Since the referrer links were created specifically for the simulation, they were defined randomly. As for the geographic locations, only some countries were used as proof-of-concept, and the probabilities were also defined randomly. Nevertheless, all these configurations are totally editable to any desired set of categories and distributions. Notice as well that this method can be applied to any number of characteristics desired, and not only the ones described.

### 3.3.2 Invalid Traffic Generation

The process is more complex for clicks marked as being invalid, since different characteristics must be emulated. Given that click fraud attacks are going to be emulated, the first step is to define the type of attack. The different attacks being emulated are defined once more by a probability distribution. However, the same type of attack can have many variations. To account for this, each type of attack has a set of attack profiles (or attack sub-types). Each attack type also has a probability distribution for its profiles. These profiles are defined by a set of characteristics. Estimating each attack's probability is not easy because their exact distribution in the "real world" is unknown. The same thing happens when defining the profiles of each attack, since the nuances of each attack are also not known. Therefore, these properties were defined based on information gathered during the study of the click fraud problem. This study revealed the common used techniques, and their characteristics.

For each click marked as invalid, a type is given. The invalid clicks are then grouped in smaller subsets containing all the clicks from each attack type. For each of these attack subsets, different attacks are then emulated. Each of these emulated attacks is generated using a randomly chosen attack profile of the main attack type. An attack profile has a set of characteristics that were defined based on common known characteristics of each attack type.

Table 3.1: Used attack types and their probabilities.

| **Technique** | **Probability** |
|---|---|
| Technique | Probability |
| Single Person | 0.1 |
| Click Farm | 0.2 |
| Affiliated Click Fraud | 0.2 |
| Botnets | 0.5 |

The configurations used for the attack types and their characteristics can be seen in Table B.1. The characteristics that are defined for each attack profile are described in Table 4.2. The sets of profiles created for each click fraud attack type can be found in Appendix A.

Table 3.2: Characteristics that define each attack profile.

| **Characteristics** | **Value Type** |
|---|---|
| Probability | Real |
| Number of Clicks (Attack Size) | Integer |
| Duration (minutes) | Integer |
| Single OS | Boolean |
| Single Browser | Boolean |
| Single Country | Boolean |
| Single IP | Boolean |
| Single Referrer | Boolean |

For each attack subset, the generator starts by picking a random profile according to the probability distribution. Then the number of available clicks in that Attack Set is decremented by the amount of the profile's attack size. This process is repeated until there are no available clicks in the attack subset. In the last iteration, a 10% margin is given to choose the last profile. If the number of available clicks plus 10% is greater than the candidate attack profile size, another attack is formed. This prevents the number of clicks of that attack type to be altered significantly when creating the different attacks. In the end of this process, the each attack subset was now divided into multiple attacks, each one with its own profile. Now that all the different attacks to be emulated are defined, the attack emulation process begins.

First, it is important to notice that the invalid clicks are not correctly distributed in time (they do not reflect the attacks' time distribution characteristics). Since all clicks were generated using the same event generator, they have the exact same time distribution as the valid traffic. Thus, the only thing differentiating them is essentially the label specifying the type. To fix this, the set of invalid clicks is used as a pool of attack seeds. Then, for each attack being emulated a click is randomly chosen from the invalid click set. This click will be the seed of the attack being emulated. All the clicks of the invalid click set are then discarded, excepting the seed clicks. Then, starting from the time instant of each seed click, the respective attack is emulated and "injected" in the valid traffic.

Each seed click is characterized using a stochastic procedure, and serves as the base for all the other clicks of the respective attack. The OS and Web browser of each seed click are drawn from the probability distributions described before. This is based on the assumption that common attacks will most likely have the same OS and Web browsers' distributions as valid traffic. For instance, if a given OS or Web browser has more people using it, it's a better target for being exploited. Common attack types will also have geographic origins usually different from normal traffic's geographic origins. This can be observed in the click fraud "heat map" in Figure 3.4. This heat map enabled the creation of a probability distribution for the geographic origin of invalid traffic. Thus, the geographic location of each seed click is randomly chosen according to this "heat map" probability


Figure 3.4: Click Fraud Heat Map (April 2008)

distribution. The IP address is then generated accordingly, as described above. Another common factor of click fraud attacks is the attackers won't use the ad links in the same proportions as the valid traffic's. The reason lies in a frequent motivation for the attacks. Many times the fraudsters want to increase some partner's revenues by clicking its ad links. Obviously, these partners' ad links are not known. Hence, the referrer for each seed click is randomly picked from the available categories, with equal probability. After each seed click is defined, each attack is emulated.

Using each attack's profile and seed click, the clicks belonging to each attack are simulated and then merged with the valid traffic clicks. First, some randomness is used to generate each attack's size (nClicks) and attack's duration (number of minutes the attack lasts). This prevents the attacks of having "exactly" the amount clicks and attack duration of the profiles. These numbers are generated using two Normal distribution number generators. The mean value of one distribution is the attack's size and the mean value of the other, the attack's duration. The standard deviation for each distribution is 10% of the corresponding mean value. After both values are generated for each attack, the simulator generates exactly nClicks, and spreads them across the time interval $[\text{seed click timestamp}, \text{seed click timestamp} + \text{attack duration}]$.

All clicks of the attack being emulated start by having the same characteristics of the seed click, except the timestamp. Depending on the attack profile characteristics, these new clicks' characteristics will be either maintained or modified. This is done using the Boolean characteristics described above. For instance, if an attack's profile has "Single OS" set to true, all the clicks of that attack will have the same OS as the seed. On the contrary if "Single OS" is set to false, a random OS will be chosen for each click of the attack. This is applied to all the Boolean characteristics. After this process is terminated for all clicks of all attacks, the traffic simulator has the final set of clicks that will be used to generate the dataset.

### 3.3.3 Click Generation

The final set of clicks containing both valid and invalid traffic is outputted in the form of a simulation log file. This log file contains the information about each click, including its type (valid or invalid). The log file is then used to perform the real clicks. Each line of the log file corresponds to a click, and has the click's characteristics. These are the timestamp, OS, Web browser, IP address, geographic location, ad referrer and click type (valid or invalid). For those clicks that are invalid the attack type, profile name and attack ID are also contained in the log file. The attack ID enables identifying exactly the clicks from each attack, in case of overlapping attacks from the same attack type and profile.

The final step of the dataset generation process is performing the clicks and collecting the data. A click generator designed for the purpose reads the simulation log file and performs the clicks on a specially designed web site. Each click is performed on the OS and Web browser specified in the log file. The remaining attributes are saved on the cookie data, to be processed later. A website is specially designed for the purpose. This web site contains all the ad links used in the simulation, which all lead to the same web page. This web page contains the call to the AuditService data collection module, also known as JavaScript Interaction Code (JIC). For each click performed by the click generator, the information about the click is gathered and saved in the AuditService database. A more detailed description of the data collection process and collected click attributes can be found in Section 2.5.5.

### 3.3.4 Preliminary Preprocessing

Once the data is all gathered and saved in the database, some preliminary preprocessing steps have to be done. As mentioned before, each click is made using the correct OS and Web browser for each click. However, the remaining information is passed using the cookie. There are several reasons for this. The first is that the simulation time is different from the time at which the clicks are performed. Plus, the real IP addresses and geographic location used by the click generator are different from the IP address and geographic location present in the simulation log. Finally, the information about the validity of the clicks and the attacks also had to be saved in the database for the validation process. So, the first preliminary preprocessing step was parsing the cookie data of each click to define its simulated characteristics.

The simulation information saved in the cookie was processed directly on the database. Given that the cookie is a text string, the extraction of the several simulation attributes was made using regular expressions. Using SQL operations the several attributes were extracted and their respective columns on the database were updated. Another important preprocessing step was to put the data from the clicks into a format more suitable for the mining process. This is, a single table where each row corresponds to a data point, and

each column to an attribute of the data point. The reason for this was that the AuditService data model saves the information about each click across multiple tables. To solve this problem, several stored procedures were created to dynamically gather all the data relative to all clicks and form a single table. Another important aspect of preprocessing was the treatment of the userAgent attribute ( string which contains information about the click's Web browser, OS and language). Using a regular expression, only the Web browser name was extracted, ignoring the version. The version was ignored, because when simulating the traffic the web browser versions were not taken into account. Thus, since the click generator used different machines with different Web browser versions, this could affect the analysis. The OS and language data were ignored since this information was already contained on other attributes.

### 3.3.5 Relevant Features

[KTP+09] defines a click fraud penetration testing system as:"a system that provides an environment for testers to conduct experimental attacks on a click fraud detection system, with the main goal of proactively finding vulnerabilities in the system before being exploited by malicious attackers." The traffic simulator described here is similar to a click fraud penetration testing system in many aspects. The main difference is that this traffic simulator was designed with the main goal of validating a click fraud detection approach. Even so, some of the recommendations presented by [KTP+09] make perfect sense and thus are valid for the traffic simulator. Thus, some of these recommendations were applied. These were: Isolation (does not influence real advertisers, publishers, and users), Resource Emulation (provide virtual resources to conduct more powerful attacks), Virtual Time Compression (speed up simulation time of attack) and Traffic Composition (complete control over the traffic). The way in which these recommendations were followed will be described briefly in the following paragraphs.

- *Isolation*: The generated traffic does not influence any real advertiser, publisher or user. The environment is completely simulated, and thus the traffic generation doesn't affect any real entity.

- *Resource Emulation*: The generated clicks emulate real resources (OS's, Web browsers, IP addresses, geographic locations and ad links). This way it is possible to emulate a virtually infinite number of different machines and users. In the future, the number of emulated resources can be even larger. Examples of other resources that can be emulated are OS languages, screen resolutions and installed web browser plugins. It is easy to see that the more resources are emulated, the more realistic the generated traffic will become.

- *Virtual Time Compression*: The simulation time is shorter than the time recorded in the data. Each traffic simulation is performed between an arbitrary time period. This time period is defined by a start and end dates, completely configured by the user. It enables the simulation of long periods of time, in a much shorter real period of time. For instance, a simulation log file for 1 month of traffic may take some minutes to generate. Then, performing the clicks may take some days, depending on the available resources. The result is 1 month of data generated in some days. All simulated clicks have a virtual timestamp passed on the cookie. Each virtual timestamp is then retrieved from the cookie and updates the real timestamp of each click. Each click is then converted to this "virtual time". This is a virtual time compression method.

- *Traffic Composition*: Even though stochastic methods are being used for generating traffic, the traffic composition can be completely controlled. One has total control of the generated traffic by:

  - Defining the virtual time period of the simulation.
  - Defining the average visit rates of the traffic.
  - Defining the categories for each click characteristic being generated (e.g. simulated OS's, Web browsers, etc).
  - Defining the probability distribution of each characteristic.
  - Defining the probabilities for each type of traffic.
  - Defining the probabilities for each type of attack.
  - Defining the several profiles for each type of attack and their probabilities.

## 3.4 Statistical Analysis

In this methodological approach, the statistical analysis has the purpose of modelling the normal traffic distribution, to then detect deviations (outliers) from it. The idea is that these outliers may reflect click fraud attacks. From studying most common attack types and their characteristics, one can conclude two things. First, larger scale click fraud attacks (such as botnets) are likely to cause a sudden increase in the number of visits of a particular site. And secondly, all the elements of an attack tend to have some characteristic in common. This can be due to technical or resource limitations (e.g. exploiting a vulnerability of an operative system (OS) or web browser, using the same IP address, etc), or because of the motivation (e.g. using the same ad referrer by clicking the ads of a partner website to increase its revenues). Thus, some attacks will likely cause deviations in normal traffic patterns/distributions not only in the number of visits in a given time period

but also on other dimensions, such as the number of visits of a given OS, web browser, etc. Even so, there are obviously more sophisticated attacks that don't cause these deviations. These attacks are able to merge perfectly with the normal traffic pattern. The cause may be: the attack lasts a long time (doesn't cause a sudden traffic spike); the attack is too heterogeneous to cause any substantial deviation on another dimension; or both. Any attack having these characteristics can't be detected by this approach. However it is also not the purpose of this approach to detect all types of click fraud attacks. In this thesis it is proposed an approach that is able to detect those types of click fraud attack that cause deviations in the website's normal traffic patterns.

To accomplish the task of detecting these attacks, the base traffic distribution (pattern) must first be modelled. Using this base distribution, those observations that deviate too much from it are labelled as outliers. These outliers are then considered suspects of having been caused by a click fraud attack.

In the context of fraud detection, local outliers describe observations that are anomalous when compared to subgroups of the data. On the contrary, a global outlier is an observation anomalous to the entire data set. [Bolton:2001] states that local outlier detection is effective in situations where the population is heterogeneous. This is the case for web traffic data. If we can identify the traffic pattern of a particular time period (e.g. number of visits from 13:00 to 14:00), then a different observation of the same time period is a local outlier if it is anomalous to the traffic pattern of that time period. Even so, it is not necessarily anomalous to the entire population of different time periods. For example, a total of one thousand visitors in an hour where, historically, the number of visitors has been under one hundred clicks might be considered as a local outlier. However, such an observation may not have been considered unusual if it had occurred in a high traffic hour, and thus would not be a global outlier.

### 3.4.1 Modelling the Base Distribution

In web traffic data, the traffic characteristics in a given time period can vary according to a smaller time unit. This smaller unit can be the specific time of the day, week, month, etc. For instance, the number of visits in a given website may vary according to the hour of the day. To accurately model the base traffic distribution of this time period, these smaller time units with different traffic characteristics must be identified. Therefore, each of these time periods can be seen as having its own local traffic pattern. And so, the set of all these local traffic patterns is what forms the global traffic pattern.

Considering that the traffic characteristics may vary according to a specific time unit, they are also likely to repeat over time. This can be seen as a cyclic pattern. For example, if the number of visitors in a given website between 13:00 and 14:00 is 200 visitors in a certain day, it is likely that between 13:00 and 14:00 of the next day the number of

visitors will be similar. It is obvious that there will be exceptions. For instance, if a given website announces a new product or service, the number of visits is likely to increase when compared to previous observations corresponding to the same time. Even so, with a sufficiently large number of observations, a pattern will emerge.

The central limit theorem allows the statement:

> "The distribution of a sum or average will tend to be Normal as the sample size increases, regardless of the distribution from which the sum or average is taken."

Based on the central limit theorem, it is assumed that with enough observations the total number of visits in a given time unit will approximately follow a normal distribution. So even if the underlying distribution doesn't follow a normal distribution, the sum will. This is the foundation of the entire statistical process being described here. It is easy to see that this process can be extended to any other attributes of the traffic that can be summed. By doing so, patterns on other attributes can be detected, such as the number of visits of a given OS, web browser, geographic location or referrer. Since these attributes are usually categorical variables, combining all the categories of each dimension results in a histogram. These histograms reveal the traffic patterns relative to the distribution of each attribute in a given time unit. Therefore, the base traffic distribution for the several dimensions (attribute value or category) of each time unit is modelled as being approximately normal, by using the sums of each dimension. Then, all the base traffic distributions of each time unit together form the global base traffic distribution.

To reflect the cyclic traffic behaviour and analyze the data using these smaller time units, a certain degree of overlapping has to be applied to the analysis time period. Doing so, these cyclic traffic patterns emerge and a sufficiently large number of observations are achieved. The analysis time period is defined by a start and end dates. For the overlapping to be possible, this time period has to be split using two smaller time measures. The first, called the Time Window (TW), is responsible for the overlapping factor, splitting the analysis time period into several time slices. The second, called the Time Unit (TU), divides each TW into the smaller time units that contain the local traffic patterns.

The TU is the real comparison unit between different TW's, which allows the approximation to the normal distribution. This approximation enables the detection of outliers using statistical tests designed for normal data. Since the TU has a configurable length, it creates other important advantages. It allows a more precise analysis, by comparing shorter periods of time each time and by "potentially" isolating more the existing attacks. Also, when an outlier is detected it will correspond to a smaller slice of time and thus a smaller dataset.

For a more intuitive analysis, the TW's always start at the established start of the chosen time measure, i.e a Week time window will start at Sunday, a Day time window

will start at 00:00, etc. This way the obtained time window data is more "standardized". Following the same criteria as the time window, all the time units start at the established first instant for the unit (e.g. hours always start at HH:00). As a result of this, the first and last windows may be incomplete due to the start and end dates. Obviously, the first TU of the first TW may also be incomplete, as well as the last TU of the last TW. This behavior can easily be modified to use the start date as the start of the first TW and first TU(even so, the last TW and last TU could be incomplete)

Consider the analysis time period $P$. After defining the TW length, the time period $P$ is divided into $n$ time windows $W_i, \quad i \in 1 \ldots n$:

$$P = W_1, W_2, \ldots, W_n$$

Afterwards, the TU length is defined and each TW is divided into $k$ equal length units $U_j, \quad j \in 1 \ldots k$. Considering the case where all TW's are complete, an arbitrary TW $W_i$ can be defined as:

$$Wi = U1, U2, \ldots, Uk$$

And where each element is defined as $W_i U_j, \quad i \in 1 \ldots n, j \in 1 \ldots k$

So, $P$ can now be defined as:

$$\begin{aligned} P = \{&W_1U_1, W_1U_2, \ldots, W_1U_k, \\ &W_2U_1, W_2U_2, \ldots, W_2U_k, \\ &W_nU_1, W_nU_2, \ldots, W_nU_k\} \end{aligned}$$

To proceed with the statistical analysis, all the time units with the same index are grouped and form an Analysis Subset (AS). There will be as many Analysis Subsets as different Time Units. For each of these subsets, the base distribution will be modelled using several dimensions as described above. Once this step is complete the statistical test for the outlier detection may be applied.An Analysis Subset $AS_j, \quad j \in \{1..k\}$ can be defined as:

$$AS_j = \{W_1U_j, W_2U_j, \ldots, W_nU_j\}, \quad j \in \{1..k\}$$

Figure 3.5 illustrates the time division process and how the data is grouped for the analysis. This is a case where both first and last windows are incomplete.

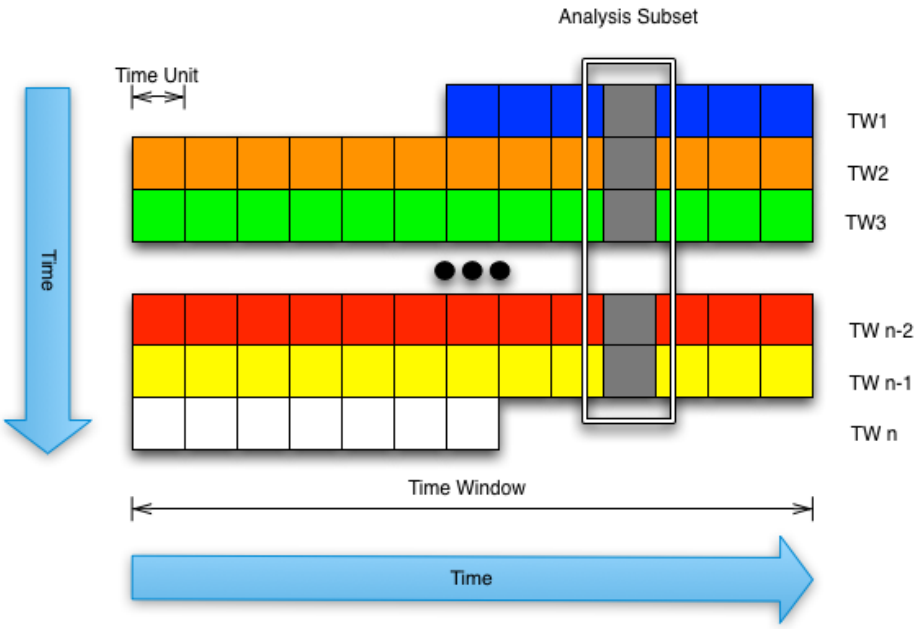


Figure 3.5: Time Division process.

### 3.4.2 Detecting Deviations (Outliers)

Once the Analysis Subsets are formed, the following step is to label suspected outliers for further study. For normally distributed data, three different methods are available: z-score method, modified z-score method, and boxplot method [BD93, BL84]. These techniques are based on robust regression methods. All of the experimental observations are standardized and the standardized values outside a predetermined bound are labelled as outliers [RL87]. Since the boxplot method requires a graphical representation it was not considered as an option, because one of the requirements of the approach is to perform automatic detection.

In a z-score test, the sample mean and standard deviation of the entire set are used to obtain a z-score for each data point, according to following formula:

$$Z_i = \frac{x_i - \bar{x}}{s}, \quad s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

Figure 3.6 illustrates the several existing relationships in the Normal Distribution. In this context, the more relevant are the cumulative percentages and the Z scores. A test heuristic states that an observation with a z-score greater than 3 should be labelled as an outlier (which in fact means it is not contained in the interval $\bar{x} \pm 3\sigma$). However, this method is not a reliable way of labelling outliers since both the mean and standard deviation are affected in the presence of outliers. The modified z-score was developed

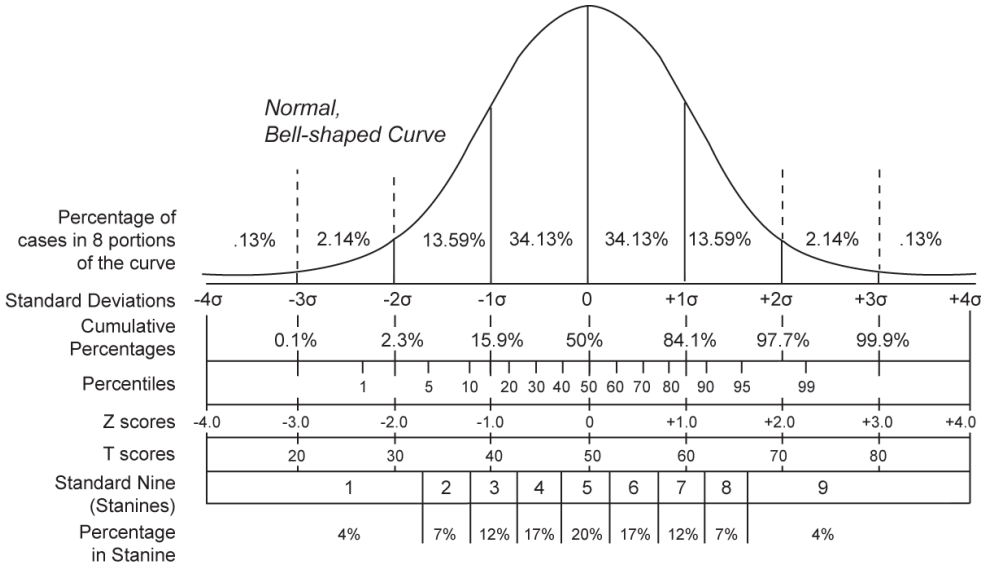


Figure 3.6: Normal Distribution and Scales.

to avoid this interference while using the test's theoretical basis. In a modified z-score test the z-score is determined based on outlier resistant estimators. The median and the median of absolute deviation about the median (MAD) are such estimators.

MAD is calculated and used in place of standard deviation in z-score calculations, as can be described by the following steps:

- Calculate the sample median,
- Calculate the MAD about the sample median, using:

$$MAD = median(|x_i - median(x)|)$$

- Calculate the modified z-score for each observation, using:

$$Z_i = 0.6745 \times \frac{x_i - median(x)}{MAD}$$

- If an observation has $|Z_i|$ greater than a specified threshold, it is labelled as an outlier.

The test heuristic states that an observation with a modified z-score greater than 3.5 should be labelled as an outlier. This is a reliable test since the parameters used to calculate the modified z-score are minimally affected by the outliers. The probability of a given value having a z-score greater than 3.5 is 0.000233. This is means it is not contained a $\sim$99.98% confidence interval. This assures that those values marked as outliers are with great probability true outliers. Even so, during experiments several values for the z-score

threshold were used. For instance, lower values are able to detect "milder" outliers, which could correspond to potential click fraud attacks.

The modified z-test is applied to all the Analysis Subsets, and those Time Units detected as outliers are saved for further study. This test can be applied whenever the data is normally distributed. Since the base traffic distribution for the several dimensions (attribute values or categories) of each time unit is modelled as being approximately normal, this outlier detection test can be applied.

The outlier detection can only be applied to those dimensions for which the base distribution was modelled. Currently it is applied to 5 dimensions: the total number of visits, number of visits per OS, number of visits per web browser, number of visits per geographic location and number of visits per referrer. These specific dimensions were chosen because are some of the common characteristics attacks have in common. Since this is a proof-of-concept, this small number of dimensions was considered to be enough to demonstrate the methodology capability of detecting outliers.

With the exception of the first dimension, all attributes are categorical variables. As said before, combining all the categories of the same attribute results in a histogram. These histograms provide the information about the distribution of that attribute. Since each bar of the histogram (corresponding to a given category), follows an approximately normal distribution across all the other bars of the same category, this test can also be applied. Instead of only being tested for outliers based on each category's sum, these categorical attributes are also tested based on the frequency of each category. This frequency analysis gives the system more robustness. For example, a sudden increase in the number of visits of a given categorical attribute (all categories together) may not reflect a deviation in the frequency of these categories. This may indicate that it was just a normal traffic oscillation, and not an abnormal spike. Otherwise, the system could detect outliers in all the categories of the attribute, since it was ignoring that the distribution of these categories was perfectly normal.

For each Analysis Subset, the outliers are detected in all dimensions. The result is the set of Time Units on which outliers were detected. Each of these Time Units will contain details about every dimension where an outlier has been detected, in that specific time instant ($W_iU_j$). These Time Units are from now on considered Traffic Outliers, having one or more characteristics (the outliers of the different dimensions). An example of a

Traffic Outlier (TO) containing a single characteristic for each dimension could be:

$$
\begin{aligned}
TO(W_iU_j) = \{&\{\text{Total number of clicks dimension :} \\
&\quad (200 \text{ total extra clicks})\}, \\
&\{\text{OS dimension :} \\
&\quad (50 \text{ extra Windows 7 clicks}, 10\% \text{ extra in frequency})\}, \\
&\{\text{Web Browser dimension :} \\
&\quad (190 \text{ extra Internet Explorer clicks}, 20\% \text{ extra in frequency})\}, \\
&\{\text{Geographic Location dimension :} \\
&\quad (205 \text{ extra China clicks}, 30\% \text{ extra in frequency})\} \\
&\{\text{Referrer dimension :} \\
&\quad (10 \text{ extra adreferrer 1 clicks}, 2\% \text{ extra in frequency})\}\}
\end{aligned}
\tag{3.1}
$$

By looking at this example, one could easily see that valuable information about the traffic deviations' characteristics is obtained from this outlier detection methodology. For each detected TO, those characteristics that pass the Minimum Frequency Deviation threshold are then used to characterize the suspect attacks.

## 3.5 Outlier Analysis (Attack Characterization)

Each detected Traffic Outlier (TO) is then analyzed individually. The objective of this analysis is to characterize possible click fraud attacks. This attack characterization will enable an accurate separation of the clicks that caused the traffic deviation from those belonging to the base traffic pattern. Each TO consists of a specific time interval of the data (TU belonging to a TW), containing a set of outlier characteristics. When analyzing these outlier characteristics, there are 3 possible scenarios, summarized as follows:

- The outlier is too heterogeneous, and it is either a false positive (contains no attack), or reflects an undetectable attack (using the current dimensions).
- The outlier contains a single attack.
- The outlier contains more than one attack.

In order to detect the correct scenario for each case, the different characteristics that constitute each outlier are analyzed and compared. Depending on the relations they have between them, the current scenario is defined. The definition of the current scenario directly results in the characterization of the possible click fraud attacks. This process is

done in three steps, which can be seen as three filters. Each step detects (and filters out) one of the above-mentioned scenarios, and thus characterizes the possible attack(s).

The first step is to check whether the outlier is too heterogeneous and therefore is either a false positive or an undetectable attack. An outlier is considered too heterogeneous when it only causes a deviation in the total number of visits. In these cases, there is not enough information to allow the accurate differentiation of the clicks that caused the deviation from the ones belonging to the base distribution. This observation can be the reflex of either a normal traffic oscillation, or of a more sophisticated attack. Thus, if a given Traffic Outlier only has a deviation in the total number of visits, it is discarded from the rest of the analysis.

The second step tries to identify if the TO contains a single attack. To do so, the several dimensions are compared to check if their deviations have some degree of similarity. First, the maximum amount of extra clicks is extracted from the set of existing outlier characteristics. This maximum amount if considered the "estimated" attack size. Then, the number of extra clicks of each characteristic is compared to this estimate, measuring their similarity. If a given characteristic passes a certain threshold of similarity with the attack size estimate, it is likely to be a characteristic of the attack itself. This threshold is known as Dimension Similarity. This dimension similarity is the absolute deviation of the dimension's amount of extra clicks relative to the estimated attack size, and is calculated as follows:

$$\text{Dimension Similarity} = \frac{|\text{dimension extra clicks} - \text{estimated attack size}|}{\text{estimated attack size}} \tag{3.2}$$

The result gives the deviation in terms of percentage relative to the estimated attack size. Currently the threshold is defined so that dimensions with a maximum of 0.15 (15%) of absolute deviation are considered for analysis. This enables a good matching, leaving some room for possible estimation errors.

The set of characteristics that pass this threshold are grouped and considered to characterize the suspected single attack.

Consider the example described in Expression 3.1. One can easily observe that it would be most certainly a single attack, with an estimate size of 205 clicks. Besides having as a characteristic Geographic Location = China (since it was the maximum), it would also have Web Browser = Internet Explorer, because of the similarity in the amount of extra clicks ($\frac{|190-205|}{205} \approx 0.7, 0.7 < 0.15$). Both the OS and Referrer dimension characteristics would be discarded, because they passed the Dimension Similarity threshold ($\sim 0.26$ and $\sim 0.95$, respectively).

The third and final step is designed for those TO's that in the previous step weren't classified as a single attack. In other words, none of the outlier characteristics had a

deviation sufficiently similar to the estimated attack size (which in this case would be the total number of extra clicks). Given this, these TO's are now treated as containing more than one attack. Each TO is then analyzed in search for the outlier dimension that accurately "divides" it into multiple attacks. This specific dimension will enable the characterization of each of these multiple attacks. This dimension is the one that has more characteristics and, at the same time, the sum of each of these characteristics' extra amounts of clicks is similar to the estimated attack size.

Once the different attacks have their main characteristic identified, the remaining characteristics are analyzed to see if they are likely to also define one of these attacks. Following the approach described in the previous step, these characteristics are grouped according to the degree of similarity they have to the estimated attack size. The only difference is that the similarity is this time measured against the estimated attack size of each attack. Once this procedure is terminated, each attack is treated independently according to its own characteristics. The estimated size of each attack is recalculated, using the average of all the attack's characteristic's extra clicks. This provides a more accurate estimate of the attack size for the next steps.

Consider the following TO example:

$$\begin{aligned} TO(W_iU_j) = \{&\{\text{Total number of clicks dimension :} \\ &\quad (460 \text{ total extra clicks})\}, \\ &\{\text{Web Browser dimension :} \\ &\quad (300 \text{ extra Internet Explorer clicks}, 25\% \text{ extra in frequency})\}, \\ &\{\text{Geographic Location dimension :} \\ &\quad (200 \text{ extra India clicks}, 14\% \text{ extra in frequency}), \\ &\quad (100 \text{ extra United States clicks}, 7\% \text{ extra in frequency}), \\ &\quad (150 \text{ extra Russia clicks}, 11\% \text{ extra in frequency})\}\} \end{aligned} \tag{3.3}$$

In this case, the analysis would conclude the TO contains 3 attacks originating from each one of the 3 countries. This outcome is due to the Geographic Location dimension having the maximum number of characteristics (3), and the sum of these characteristics' extra clicks being similar to the estimated attack size (sum of 450 clicks vs. estimated attack size of 460 clicks). Then, it is easy to see that the Mozilla Firefox characteristic would be grouped with the United States sub-attack, since the extra amount of clicks are very similar between them (95 clicks vs 100 clicks, which resulted in a 0.05 Dimension Similarity).

After all the TO's are analyzed, the datasets are retrieved for those TO's on which attacks were detected. The datasets consist of the clicks contained in the time intervals cor-

responding to the TO's ($W_iU_j$). For those TO's on which multiple attacks were detected, the datasets are divided into multiple smaller datasets according to the dividing attribute's value. Using the previous example, the TO dataset would be split into 3 datasets, one containing only clicks from India, other containing only clicks from the United States and one with all the clicks from Russia. The attacks' datasets are then preprocessed to prepare the data for the clustering analysis.

## 3.6 Preprocessing

Once the dataset of each attack is obtained, some extra pre-processing has to be made before running the clustering algorithm. This consists of several steps, which are:

- *Select Attributes*

  The choice of the attributes used for the analysis is defined using the attack characteristics. The attack characteristics (except the total number of visits) directly relate to the dataset attributes and respective values. So, each of these characteristics can be seen as a pair $attribute = value$. Since the attributes that define the attack are known, these are the selected attributes for the clustering analysis. This increases the probability of the attack being isolated into a single cluster, given that other attributes don't interfere in the clustering process.

  This simplistic attribute selection criterion is only to be considered as a proof-of-concept. It demonstrates that the outlier analysis results can be used to optimize the chosen attributes for the clustering algorithm. In a real scenario, these attributes would possibly be given a greater weight and any other attribute found relevant to the analysis would have a normal one. This new approach would give more importance to the attributes found in the outlier analysis, but wouldn't discard others. It could potentially isolate even more each attack, by revealing hidden patterns on other attributes.

- *Replace Missing Values*

  When a given attribute has some elements with no data, the attribute is said to contain missing values.

  Because clustering algorithms normally can't handle missing values properly, this step is used to enable all attributes to be used even if they contain missing values.

  In the generated dataset used for the experiments, only the attribute containing the user OS had missing values. Since the dataset generation process creates a simulation log file, it was possible to verify that every missing value corresponded to "Windows Vista". Apparently, some browser implementations for this operative system failed to provide this information and thus this attribute was empty.

- *Remove Correlated Attributes*

  Used as a precautionary measure, the feature of removing possibly correlated attributes was added to the workflow. Correlated attributes have dependencies between them, which negatively affect the clustering process. Using this filter, when two attributes have a correlation factor greater than a configurable threshold (currently 0.95), one of them is discarded from the analysis.

- *Remove Useless Attributes*

  Another precautionary measure was the use of a filter to remove useless attributes. An attribute is considered useless when it has a constant value, and thus brings no information to the analysis. In fact, such attributes can even worsen the results if used. Also, as a consequence of removing these unnecessary attributes, the number of attributes is also reduced. Fewer attributes require fewer resources for the clustering algorithm.

- *Convert Nominal Attributes to Numerical*

  Even though some algorithms don't require all attributes to be numerical (such as DBSCAN), in order to perform normalization all attributes must be discretized. Because of this all attributes are subject to a discretization before being normalized.

- *Normalize Attributes*

  To prevent some attributes to have a greater influence on the analysis, these were all normalized using the z-transformation method (also called z-score normalization). This method was chosen since it's more robust in the presence of extreme values (outliers). The normalization process causes that all attributes have their values in the same range of values, thus the same scale. Using the z-score normalization, each normalized value is computed as:

$$x' = \frac{x - \bar{x}}{s}$$

  After these preprocessing steps each dataset is then used as input for the clustering algorithm, which will attempt to isolate the clicks that form each attack.

## 3.7 Clustering Analysis

Clustering techniques are used to separate the clicks responsible for the deviation from those belonging to the underlying traffic pattern. A specific attribute configuration is defined for each attack's dataset, using the results from the attack characterization process. The clustering analysis is applied to each attack's preprocessed dataset. The result of

each run of the clustering algorithm is a set of clusters, called cluster set. So, for each of these datasets a cluster set is obtained. A cluster is then chosen from each cluster set, by measuring the similarity of each cluster to the attack characteristics obtained in the outlier analysis.

The chosen clustering algorithm was DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [EKJX96]. DBSCAN's definition of a cluster is based on the notion of density reachability. Basically, a point $q$ is directly density-reachable from a point $p$ if it is not farther away than a given distance $\varepsilon$ (i.e., is part of its $\varepsilon$-neighborhood), and if $p$ is surrounded by sufficiently many points such that one may consider $p$ and $q$ to be part of a cluster. $q$ is called density-reachable from $p$ if there is a sequence of points with and $p_1 = p$ and $p_n = q$ where each $p_{i+1}$ is directly density-reachable from $p_i$. Note that the relation of density-reachable is not symmetric (since $q$ might lie on the edge of a cluster, having insufficiently many neighbours to count as a genuine cluster element). So, the notion of density-connected is introduced: two points $p$ and $q$ are density-connected if there is a point $o$ such that $o$ and $p$ as well as $o$ and $q$ are density-reachable.

A cluster, which is a subset of the points of the dataset, satisfies two properties:

1. All points within the cluster are mutually density-connected.

2. If a point is density-connected to any point of the cluster, it is part of the cluster as well.

DBScan requires two parameters: $\varepsilon$ (*epsilon*) and the minimum number of points required to form a cluster (*minPts*). It starts with an arbitrary starting point that has not been visited. This point's $\varepsilon$-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized $\varepsilon$-environment of a different point and hence be made part of a cluster. If a point is found to be part of a cluster, its $\varepsilon$-neighborhood is also part of that cluster. Thus, all points that are found within the $\varepsilon$-neighborhood are added, as well as their own $\varepsilon$-neighborhood. This process continues until the cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of another cluster or noise.

The main features of this algorithm are the ability of discovering clusters of arbitrary shape, handling noise and only needing one scan on the data. Another interesting characteristics is the fact of not needing to specify the number of clusters apriori, unlike $k$-means.

These characteristics were decisive when choosing the clustering algorithm. Since we are looking for clicks that already have a known set of attribute values in common, the $\varepsilon$ parameter is set relatively small. By doing so the algorithm only clusters the data points that have many attributes in common, creating more compact clusters. Compact clusters

are likely to isolate more the attacks. This result means the used parameters can be the same for every case, which is ideal for an automated process. Ultimately this reflects on a better detection accuracy, because fewer clicks go to clusters by having fewer attributes in common. This obviously applies to the number of valid clicks that could get clustered in the chosen clusters. So, DBSCAN allows an automated clustering process that obtains compact clusters thus minimizing potential false positives in the chosen clusters.

Due to the simplified attribute selection approach, a problem arise when there was only one pair $attribute = value$ that characterized the attack. Since clustering algorithms require at least two attributes, the attacks with these characteristics could not be isolated using clustering. Experiments were made using other attributes simultaneously, but it was the same as simply filtering the dataset by that specific attribute value. In order not to lose the possibility to detect these attacks, the datasets were simply filtered using this criterion. As expected, the number of valid clicks included in the final set increased but this error was attenuated by the scoring system described in the next section.

### 3.7.1 Cluster Selection

After applying the DBSCAN algorithm a set of clusters is obtained, from which only a single cluster should correspond to the suspected attack. The task at hands is how to identify the cluster that most likely contains the attack's clicks. This is done in two steps. First only clusters satisfying a Cluster Size Similarity threshold are considered. This cluster size similarity is the absolute deviation of the cluster size relative to the estimated attack size, and is calculated using the following formula:

$$\text{Cluster Size Similarity} = \frac{|\text{cluster size} - \text{estimated attack size}|}{\text{estimated attack size}} \tag{3.4}$$

The result gives the deviation in terms of percentage relative to the estimated attack size. Currently the threshold is defined so that only clusters with a maximum of 0.2 (20%) of absolute deviation are considered for analysis. This way the system is more robust to possible estimation errors.

For every cluster that is considered, all of its elements are analyzed and its attribute values compared with the attack characteristics discovered before. Every time an element matches the attack's attribute characteristics, a counter is incremented. Once all elements have been analyzed, the total number of matching elements is compared to the estimated attack size. When comparing these two values, the same similarity principle is applied. A cluster is only chosen when it passes a given threshold of similarity. This threshold of similarity is called Cluster Similarity, and is calculated by dividing the number of matching elements by the estimated attack size, as follows:

$$\text{Cluster Similarity} = \frac{\text{number of matching elements}}{\text{estimated attack size}} \tag{3.5}$$

Then, if this value is superior to a given value the cluster is considered the one containing the suspected attack. This threshold is currently set at 0.8. It requires a good degree of similarity but allows some flexibility, accounting for possible estimation errors.

## 3.8 Scoring

For each cluster that is selected as containing a click fraud attack, a suspicion score is given to all its elements. This score can me seen as a measure of traffic quality expressed in percentage, where 100% corresponds to a completely valid click, and 0% to an undeniable case of click fraud. This scoring system is part of AuditService auditing platform's philosophy. The platform contains several independent detection modules, where each one contributes with its own score for every click. The scores from all modules are then combined and a final score is computed. By using such scoring methodology, the possible errors from one module are attenuated by the scores given by the other modules. Since this approach has as final objective the integration in the AuditService platform, a scoring system had to be developed.

To calculate the score, each estimated attack's size calculated previously is used to compare with the respective chosen cluster's size. Like mentioned before, the clusters are selected according to their characteristics (size and attribute values), which are based on the previous statistical outlier detection. This means that there is a high degree of confidence that the clusters effectively contain invalid clicks. However, many times valid clicks are clustered together with the invalid clicks. These false positives (valid clicks detected as invalid) will reflect on a larger size of the cluster. If the estimated attack size was 300 but the detected cluster had 350 elements, the system estimates that around 50 clicks of the cluster are false positives.

Even though a high degree of confidence exists that the cluster contains invalid clicks, there is no way to accurately differentiate them from the false positives present in the same cluster at this point. Since the same score bust be given to all clicks from each chosen cluster, this must be calculated taking this estimated error into account. So, depending on the cluster size deviation from the estimate a score is given to all clicks. In this approach, all clicks start with 100% score. Then, the system calculates a new score for those clicks contained on the chosen clusters.

The current score formula is quite simple and is calculated based on the similarity between the chosen cluster's size and the estimated attack's size. If the cluster size is less or equal than the estimated size, the cluster is considered to contain the entire attack and a drastic score of 0% is given. On the contrary, if the cluster size is greater than the estimate it is considered to contain false positives (valid clicks detected as invalid) and the score is

computed as:

$$Score_{\text{cluster size}<\text{estimated attack size}} = \frac{\text{estimated attack size}}{\text{cluster size}} \tag{3.6}$$

For instance, if the estimated attack size was 50 and the cluster had 100 elements, the resulting score would be 50%. Thus, this reflects the "certainty" that the solution has that those clicks are in fact invalid. Even though this scoring formula is extremely simple, it proved to be quite robust during the experiments (as can be seen in Chapter 4). It revealed to be a good way to minimize some errors that might have occurred during the process. For those cases where less certainty exists, the clicks will have higher scores. Thus any possible valid clicks contained in the clusters won't be so negatively affected.

Once all the detected clusters have their scores, the detection process has terminated and the results must be validated. As mentioned before, the type (valid or invalid) of each click is already contained in the data (due to using a generated dataset). Analyzing the final set of detected clicks, one can extract some performance metrics that can be used for validation purposes.

## 3.9 Validation

Given that the exact type of each click (valid or invalid) is known from the generated data, some important detection metrics can be calculated. These are: invalid clicks correctly marked as invalid (true positives); valid clicks incorrectly marked as invalid (false positives); valid clicks correctly marked as valid (true negatives) and invalid clicks correctly marked as valid (false negatives).

A Fraud Detection system can be seen as a binary classifier. Using the set of metrics described in 4.2, the results in a Fraud Detection system resume to:

Table 3.3: Fraud Detection System possible results.

| | Fraud | No Fraud |
|---|---|---|
| Detected | True Positive (TP) (Correct) | False Positive (FP) (Error) |
| Undetected | False Negative (FN) (Error) | True Negative (TN) (Correct) |
| Total | P | N |

Some evaluation metrics that can be derived from this matrix are:

True positive rate (TPR) or sensitivity (also called recall):

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN} \tag{3.7}$$

False positive rate (FPR):

$$FPR = \frac{FP}{N} = \frac{FP}{FP+TN} \tag{3.8}$$

True negative rate (TNR) or specificity:

$$TNR = \frac{TN}{N} = \frac{TN}{FP+TN} = 1 - FPR \tag{3.9}$$

False negative rate (FNR):

$$FNR = \frac{FN}{N} = \frac{FN}{FN+TP} \tag{3.10}$$

Accuracy (ACC):

$$ACC = \frac{TP+TN}{P+N} \tag{3.11}$$

In the click fraud detection context, the TPR represents the amount of clicks that were correctly flagged as invalid from the entire dataset. On the other side, the FPR represents the amount of clicks that were marked as invalid when in fact they were valid. The TNR indicates the number of clicks accurately marked as being valid, and the FNR represents the number of clicks that were marked as valid when in reality they were invalid. The ACC measures the overall detection accuracy, using both correctly detected cases (TP and TN).

These values can only be calculated because the real type of each click can be accessed in the simulation data present on the generated dataset. Otherwise it would be practically impossible to confirm whether a click was or wasn't fraud, as discussed previously.

These metrics enable the calculation of performance metrics and the validation of the detection process. Considering that in a real life scenario one wouldn't have the information about the type of each click, two robustness measures are used: the average invalid clicks' score and average valid clicks' score. These are averages of the suspicion scores given to the final set of detected clicks. They are defined as:

$$\text{Average Valid Score } (AVS) = \frac{\sum_{i=1}^{FP} score(\text{false positive}_i)}{FP} \tag{3.12}$$

and

$$\text{Average Fraud Score } (AFS) = \frac{\sum_{i=1}^{TP} score(\text{true positive}_i)}{TP} \tag{3.13}$$

These two metrics indicate whether the system is able to give better scores in those cases where less certainty of fraudulent activity exists, and worse scores otherwise.

## 3.10 Implementation Details

This subsection briefly describes the main aspects of the developed solution, called AuditWebUsageMiner. The level of detail in this subsection will be quite reduced since the

main focus of this dissertation is the theoretical approach. So, a summary of the initial requirements and a brief description of the architecture are given.

### 3.10.1 Requirements

The main requirements for the implementation of the approach were:

- The application should be integrated in the AuditService, under the form of a Data Processing Plug-in (DPP).
- The application had to be developed using the Java language, since it is the language in which the AuditService platform is written. This way, only the required interfaces for a DPP have to be implemented when integrating the solution.
- The application should be automated whenever possible. Ideally, it should be fully automated since the main objective of each DPP is to operate without human intervention.
- The application should implement a scoring system.
- The used click stream data should be directly obtained from the AuditService database.
- The application could only use open-source libraries.

All the requirements were fulfilled with the exception of the integration in the AuditService and a completely automated process. The integration was not possible due to some delays and time limitations. The priority was the development of a functional prototype that proved the applicability of Web Usage Mining techniques for click fraud detection. The complete automation of the process was also not achieved. The reason is that for each analyzed website, the traffic patterns may be very different. Therefore, the time division process for the statistical analysis has to be adjusted for each case. Some kind of heuristic must be developed to automatically detect the optimal Time Window and Time Unit. These unfulfilled requirements are considered future developments.

### 3.10.2 Architecture

The used architecture was based on the methodology workflow described above. The main modules were Statistical Analysis, Clustering Analysis and Scoring. The Statistical Analysis module retrieves the initial data from the database, models the base distribution, detects the outliers and analyzes them. Then, it retrieves the datasets corresponding to each detected attack. The Clustering Analysis module preprocesses the data and executes the DBSCAN clustering algorithm on every attack dataset. Afterwards it chooses the cluster containing the suspect clicks. The Scoring module analyzes each cluster and

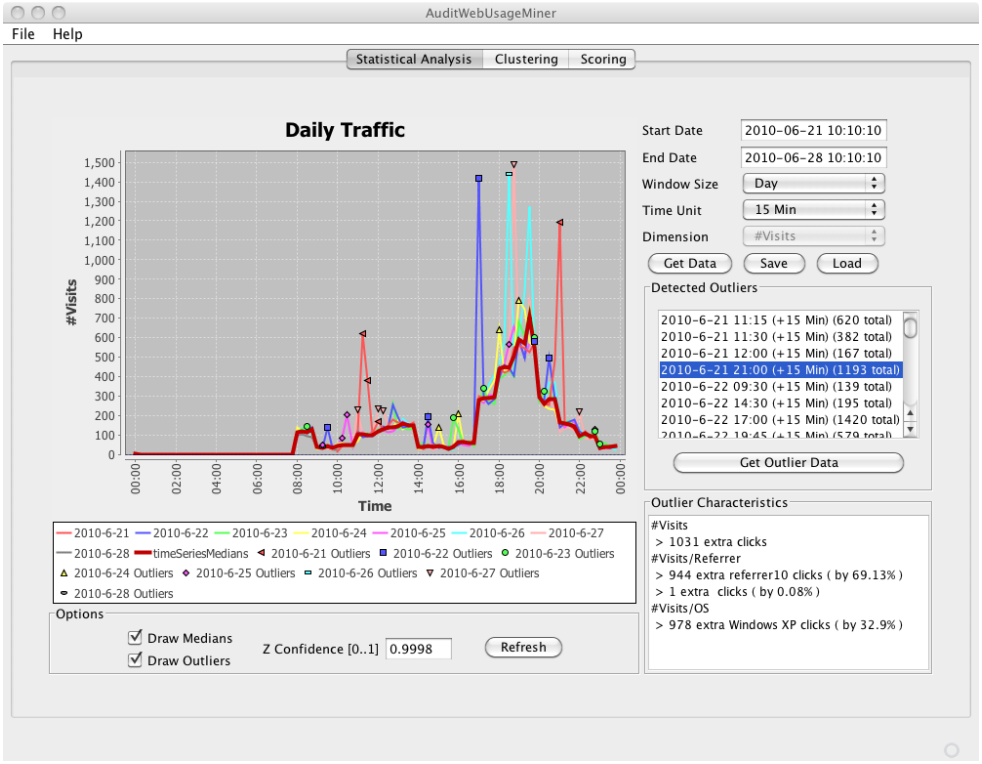


Figure 3.7: AuditWebUsageMiner Statistical Analysis Module

corresponding attack data to give a score to each detected click. Then, it computes some evaluation metrics using the traffic simulation data.

#### 3.10.2.1 Statistical Analysis Module

This module is the starting point of the whole detection process. It receives the inital parameters, and gets the data from the database. Besides performing all the statistical analysis, this module is also a powerful visualization tool. The process still requires a certain degree of human intervention. Thus, visualization plays an important role in discovering the optimal set of parameters. The visualization is made using a Time Series chart, which plots the traffic data retrieved. It reflects the overlapped windows, and shows the detected outliers. The used graphics API was JFreeChart. A screenshot of the Statistical Analysis Module can be seen in Figure **??**.

#### 3.10.2.2 Clustering Analysis Module

The RapidMiner framework was the chosen library for performing the preprocessing and cluster analysis steps. The RapidMiner API made this task easy, given its many innovative features. The intuitive process and workflow design interface enabled a fast creation and parametrization of the whole mining process. With its modular approach, it gave

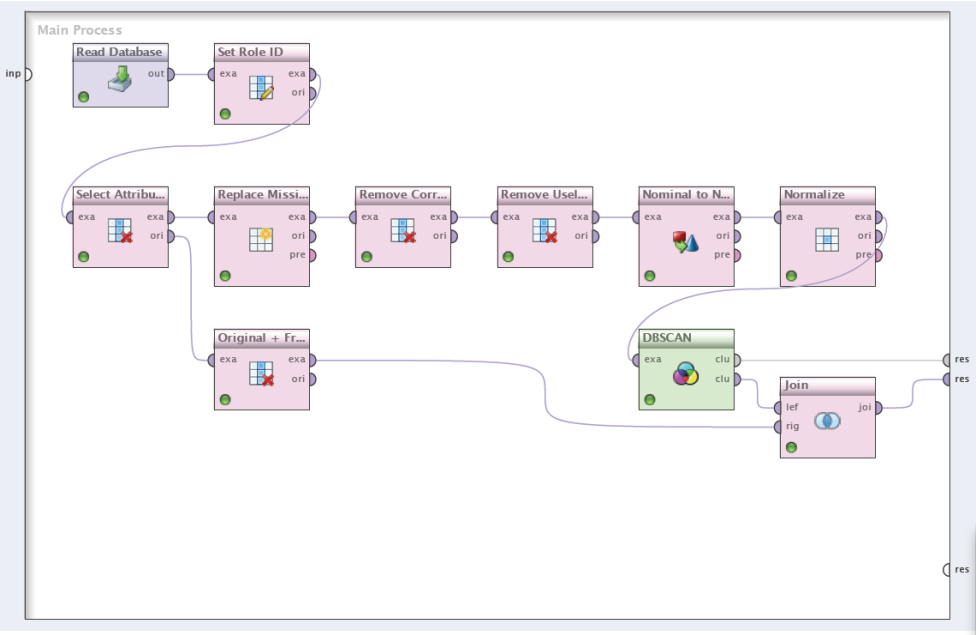


Figure 3.8: RapidMiner process workflow.

the possibility of quickly designing experiments using the most diverse configurations. Additionally, RapidMiner's result analysis functionality made easy the optimization of the process. The process was first created in the RapidMiner standalone application and then exported to the solution with some adaptations. Figure 3.8 illustrates the mining process workflow as seen from RapidMiner standalone application. A brief description of the modules will be given, starting from the left upper corner.

- *Database Read*: module responsible for retrieving the data from the database
- *Set Role ID*: module used to define the role of the "ID" attribute that identifies each click. This attribute is obviously not used in the remaining analysis, and enables a join operation performed later in the process.
- *Select Attributes, Replace Missing Values, Remove Correlated, Remove Useless, Nominal to Numerical and Normalize* were described before. The first of these modules (Select Attributes) has both outputs connected, where the one named "exa" is the dataset containing only the selected attributes and the one named "ori" the original dataset.
- *Original + Fraud*: This is a "Select Attribute" class module which does not ignore the attributes from the simulation, hence the "+ Fraud". This module serves two purposes, which are to provide the original attribute values for a better interpretation, and the simulation attributes for validation purposes. The original attributes are

needed because the preprocessing operations modify the attribute's values, making them hard to interpret in the final results.

- *DBSCAN*: the DBSCAN clustering algorithm, which outputs a cluster model and a result dataset. It also creates a new attribute containing the cluster to which each datapoint was assigned to (cluster label).

- *Join*: a join operator that uses the "ID" attribute (hence the use of the Set Role module before) and performs a join operation between two datasets. These datasets are the one resulting from the clustering algorithm and the one from the "Original + Fraud"'s "ori" output. The result of this is the dataset containing all the original attribute values and the newly created cluster label attribute.

#### 3.10.2.3 Scoring Module

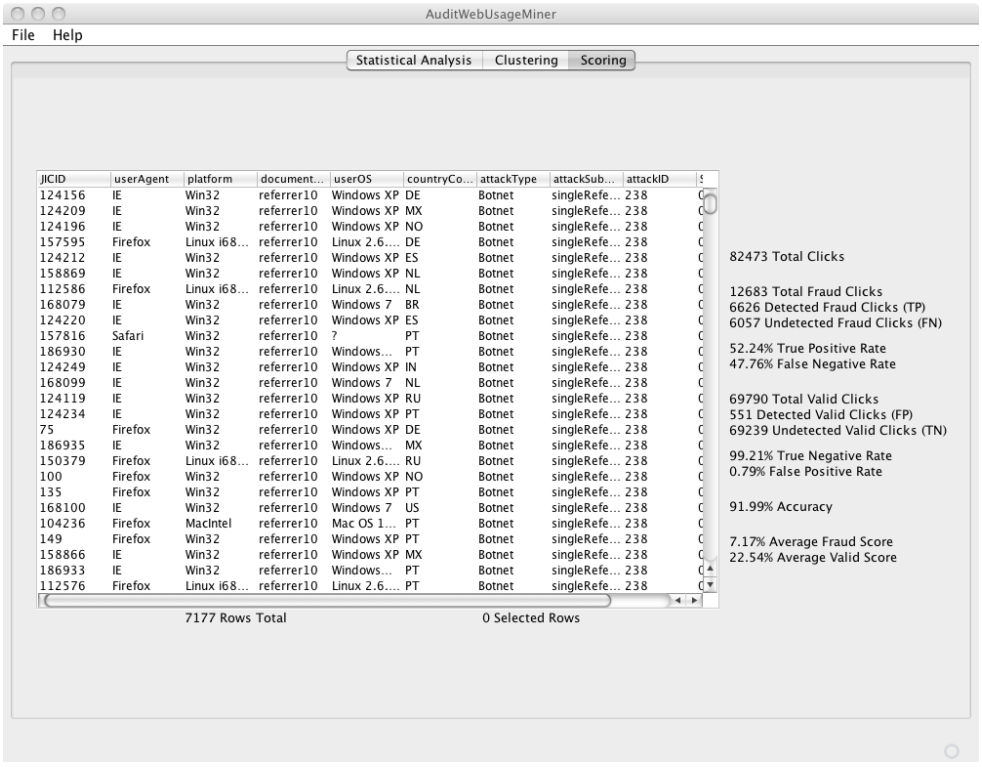


Figure 3.9: AuditWebUsageMiner Scoring Module.

The scoring module is the simpler of all three modules. It performs the score calculations, and outputs a table containing all the detected clicks, their attributes and the score. In this table the simulation attributes are also present (such as attack type, attack profile and attackID). Using these special attributes some evaluation metrics are computed, as

described previously. Figure 3.9 illustrates an example of scoring results and calculated metrics.

## 3.11 Limitations

Since this approach is based primarily on a statistical outlier analysis, the base distribution (traffic patterns) will differ from web site to web site. This distribution and consequent patterns may become visible on one website by setting the analysis time window to 1 day, while on another they become visible only when using a 1 week time window, and so on. It may even happen that the amount of collected traffic may not be enough for any traffic patterns to emerge, in which case this approach will not be effective. When no traffic pattern emerges there are only two possible ways considering the current approach. The first is to try and find the analysis time window that reveals the inherent traffic pattern of the website. If no time window reveals the traffic pattern it is possibly due to not having collected enough data, in which case the analysis should be postponed to a moment where more data is available. If both of these solutions fail, it is possible that the analyzed web site possesses a chaotic traffic pattern, which is impossible to analyze using this approach. Even so, this is unlikely considering the central limit theorem. So it can be stated that with enough data patterns will emerge under the form of normal distributions. Nevertheless, the correct analysis time window must still be used.

Another issue is scalability, because the amount of data to be analyzed can grow to be huge. Either by the web site traffic volume, or by the amount of time being analyzed. Both cases will make the analysis process require more computational resources, and therefore more time. From the statistical analysis point of view, this reflects more on the number of queries to the database and on the amount of data retrieved. Yet from the clustering analysis point of view, on the first case it will cause the detected outliers datasets to be larger and on the latter it will cause more outliers to be detected. Both scenarios will require more memory and processor resources and/or more computation time. The current implementation does not account for this problem. However, it can easily be modified to use parallel processing and even to turn it into a distributed system. Since the process is divided into steps, and we process independent sets of data at a time, it is easy to perform such improvement. When considering the database requests, one way to minimize the impact of the extra number of queries is to optimize them by grouping them, or so that they run faster and use less resources. Another measure to reduce this is to use larger time units when analyzing each time window, which will result in less queries.

As mentioned previously, the used set of attributes was small when compared to the available amount. This simplistic attribute selection criterion should be considered a proof-of-concept. In the future, the set of used attributes will surely increase. Still, if

the number of used attributes increased too much, the chosen clustering algorithm would have to be reconsidered, due to the known limitations of classic methods. A possible solution could be the use of subspace clustering techniques.

## 3.12 Conclusion

This chapter described the proposed methodology for detecting some types of click fraud attacks. These attacks can be described as attacks that have characteristics that are likely to cause deviations on the base distributions of the underlying valid traffic. These deviations are detected by the use of statistical methods. Then the deviations are analyzed, which allow the differentiation of the clicks that effectively caused each deviation. Once the clicks are detected, they are given a suspicion score. This score is based on the existing degree of confidence that those clicks in fact contain invalid traffic.

The proposed approach is considered innovative and robust. It defines a clear workflow for the detection process of some types of click fraud. It is important to state that this approach is a proof-of-concept that demonstrates the applicability of web usage mining techniques for the click fraud detection problem. The theoretical steps were described in detail for a generic scenario, but the experiments were conducted under simpler scenarios. Some extensions and adaptations may have to be made to prepare this approach for a real life scenario. The conducted experiments and the results obtained are described in the following chapter.

# Chapter 4

# Tests and Validation

## 4.1 Introduction

As mentioned before, the use of synthetic data was required for the validation process. In a click fraud detection scenario, proving the legitimacy of a given click depends mostly on the context. There is never certainty that a click is valid or invalid, when analyzing real web traffic data. Obviously, fraudsters cannot be "hired" to commit fraud and provide detailed reports of their deeds. Therefore, the uncertainty about the validity of the clicks remains. The use of synthetic data allows the comparison of the analysis result with the correct answer known previously. Hence, to evaluate the performance of the proposed approach, a generated dataset was used as input for the analysis. This dataset includes both invalid and invalid traffic. The web traffic was designed to emulate a real scenario, as described in Section 3.3.

The data between the two dates used for the traffic simulation (creating the generated dataset) constitutes then the analysis time period for the remaining process. This time period is then split into several time windows, which are then overlapped. These time windows are split once more into smaller time slices called time units. Grouping together the same time unit from all time windows forms the analysis subsets.

Once the analysis period is divided into these analysis subsets, a statistical analysis models the base distribution (normal traffic pattern) and then detects deviations (outliers) from this distribution. These deviations are now considered suspicious attacks of click fraud.

After the deviations have been detected in all time units and all dimensions, they are analyzed to characterize the suspicious attacks. Once all deviations are analyzed and the attacks characterized, this information is used to try and separate the clicks that caused the deviation from those that belong to the base distribution.

To differentiate the clicks that caused the deviation from those that belong to the distribution, a clustering algorithm is used. Each suspicious attack forms a dataset. Then,

using each deviation analysis result, the clustering algorithm is configured accordingly and is applied to the respective dataset. From the resulting cluster set, a cluster is selected if it is sufficiently similar to the suspicious attack characteristics.

For each cluster that is selected as containing a click fraud attack, a suspicion score is given to all its elements. This score can me seen as a measure of traffic quality expressed in percentage, where 100% corresponds to a completely valid click, and 0% to an undeniable case of click fraud.

When the detection process ends, the result is a set of clicks marked as being suspect of fraud, each one with a suspicion score. The final step lies in validating these results. As mentioned above, the type (valid or invalid) of each click is already contained in the data. Analyzing this set of clicks, one can know the exact amount of: invalid clicks correctly marked as invalid (true positives); valid clicks incorrectly marked as invalid (false positives); valid clicks correctly marked as valid (true negatives) and invalid clicks correctly marked as valid (false negatives). These amounts enable the calculation of performance metrics and the validation of the detection process. Considering that in a real life scenario one wouldn't have the information about the type of each click, the suspicion scores are also used as a performance metric. Since the suspicion scores are only altered for the clicks marked as being invalid, the average scores for both true positives and false positives are computed and evaluated. All the clicks that are not marked have 100% scores, and would not bring any information for this analysis.

Several sets of parameters were used for the different phases of the process, namely statistical analysis, attack characterization and clustering analysis. This chapter starts by summarizing the used metrics in Section 4.2, and then briefly describes the used dataset in Section 4.3. The used attributes are described in Section 4.4 and the different parameters, algorithms and configurations used are discussed in Section 4.5. The experimental configurations are further detailed in Section 4.6, and the results for each experiment are presented in Section 4.7. Section 3.12 contains some conclusions.

## 4.2 Metrics

The used set of metrics consists of some of the measurements described in Section 3.9. These were the two error measurements FPR (Expression 3.8) and FNR(Expression 3.10), the accuracy ACC(Expression 3.11) and the score averages AVS(Expression 3.12) and AFS(Expression 3.13). In the click fraud detection context, the TPR represents the amount of clicks that were correctly flagged as invalid from the entire dataset. On the other side, the FPR represents the amount of clicks that were marked as invalid when in fact they were valid. The TNR indicates the number of clicks accurately marked as being valid, and the FNR represents the number of clicks that were marked as valid when in reality they were invalid. The ACC measures the overall detection accuracy, using both correctly

detected cases (true positives and true negatives). These values can only be calculated because the real type of each click can be accessed in the simulation data present on the generated dataset. Otherwise it would be practically impossible to confirm whether a click was or wasn't fraud, as discussed previously.

These metrics enable the calculation of performance metrics and the validation of the detection process. Considering that in a real life scenario one wouldn't have the information about the type of each click, two robustness measures are used: the AFS (average fraud clicks' score) and AVS (average valid clicks' score). These are averages of the suspicion scores given to the final set of detected clicks. They indicate whether the system is able to give better scores in those cases where less certainty of fraudulent activity exists, and worse scores otherwise.

## 4.3 Dataset

The dataset used in the experiments was the result of a simulation generated using the traffic simulation process described in Section 3.3. The virtual time used to generate the traffic consisted of 7 days corresponding to the time between "2010-06-21 10:10:10" and "2010-06-28 10:10:10". For this simulation, the portion of invalid traffic clicks was maintained at 15%. The average number of visits over time was defined on a per hour basis. This is illustrated in Figure 3.2, where the plot of the average visit rates per hour can be seen. The used distributions for the clicks' characteristics and the fraud attack types and their probabilities were the same as described in Section 3.3. A total of 82473 clicks were generated, from which 12683 were fraud (15.37%). The fraudulent clicks had the following distribution:

Table 4.1: Generated Dataset Invalid Traffic summary.

| **Attack Type** | **Number of Clicks** | **Percentage of Fraud** | **Number of Attacks** |
|---|---|---|---|
| Botnet | 6080 | ∼47.9% | 6 |
| Click Farm | 2595 | ∼20.5% | 21 |
| Affiliated Click Fraud | 2507 | ∼19.8% | 13 |
| Single Person | 1501 | ∼11.8% | 198 |
| Total | 12683 | 100% | 238 |

Table B.1 contains the summary of the invalid traffic contained in the used generated dataset. As can be seen, the attack distribution was very similar to the one described in Section 3.3.

## 4.4 Attributes

The generated dataset contained all the attributes collected by the JavaScript Interaction Code (JIC), with the exception of those collected using Java (as described in Subsection 2.1.3.2. However, this dataset didn't contain as much information as a real traffic dataset. A real traffic dataset would contain much more information when compared with this dataset. Due to the traffic having been generated using only a total of 9 machines, from which only 4 were physical (5 were virtual machines), most of the attributes on the dataset were the same for clicks that were supposed of coming from different machines/browsers. Examples of such attributes can be the installed plugins, fonts, and screen resolutions. This obviously influenced the choice of attributes. Another important factor in the choice of the attributes was the study made on the common types of attacks and their characteristics. This revealed that most attacks have a common characteristic (at least). This fact can be due to the motivation (same referrer to increase some partner's profit), resource limitation (country of origin, IP addresses, OS's, web browsers, etc), or the exploit of a given vulnerability (platform, OS, web browser, plugin, etc). Of course, depending on the sophistication of the attacks, these common characteristics may be very well hidden, and may not even be detected with the available attributes.

Because of these two facts (low number of machines and common attack characteristics), the number of used attributes from the available amount was relatively small. These were the web browser name (extracted from the userAgent string, ignoring the version), OS name, platform, country of origin (extracted from geo IP, but in this case generated) and referrer. Even so, it must be said that this approach/framework can and will be extended in the future to take advantage of the available attributes, whenever they prove to be relevant to the analysis. Still, if the number of used attributes increased to much, the chosen clustering algorithm would have to be reconsidered, due to the known limitations of classic methods. A possible solution could be the use of subspace clustering techniques. This set of attributes was also used for the dimensional outlier analysis (with the exception of the platform), for the same reason that they are the most likely to reflect an attack's common characteristic(s).

## 4.5 Parameter Configuration

The statistical outlier detection process depends on a set of parameters that affect the detected time slices and the real amount of fraud present in them (which reflects itself on the detection quality, and therefore false positive ratio). Varying the values of these parameters helped in determining the best configuration to be used. However, some improvements can still be made in terms of automating the process. Varying the parameters also helped point out some weaknesses and strengths of the proposed approach. Each

parameter reflects a specific part of the methodology, and the specific details can be found in Chapter 3.

### 4.5.1 Start and End Dates

These are the basic initial parameters for the whole process. These define the start time and end time for the analysis, expressed in timestamp format. After these are defined, the data is collected and grouped depending on the Time Window and Time Unit parameters described below. The values used in this experiment were Start Date = "2010-06-21 10:10:10" and End Date = "2010-06-28 10:10:10", which were the parameters used in the traffic simulation.

### 4.5.2 Time Window (TW)

This parameter corresponds to the length of the main sampling window, which will divide the time interval being analyzed in several time slices that will then be overlapped to proceed with the analysis. For a more intuitive analysis, at the moment these always start at the established start of the chosen unit, i.e a Week time window will start at a Sunday, Day time time window will start at 00:00, etc. This way the obtained time window data is more "standardized", although due to the start and end dates it may result in incomplete windows. However, this behavior can easily be modified and a sliding factor can be applied to the windows. This parameter can have the following values: Month, Week, Day, Hour. In this experiment the value was fixed at Day because of the dataset characteristics (traffic pattern defined on a daily basis). In a real traffic scenario, the parameter value would have to be adjusted in order for the traffic pattern to emerge. In the worst case, no pattern would exist and this approach would fail. Even so, with a sufficiently large number of observations and the correct Time Window, a pattern will emerge. This assumption is valid due to the central limit theorem.

### 4.5.3 Time Unit (TU)

The time unit parameter defines how the time window will be split into smaller time slices. These smaller time slices will then be used to gather the data from the database for all time windows, on the specified dimensions. For instance, with a time window of 1 Day and a time unit of 1 Hour, for each window the data will be gathered and grouped by hour, for all the dimensions. After the data is collected and grouped it is analyzed unit by unit for all time windows. For instance, using the previous example, the data corresponding to 16:00 to 17:00 from all the windows will be analyzed in search for outliers independently. Following the same criteria as the time window, all the time units start at the established instant for the unit (e.g. hours always start at HH:00).

This analysis across all the time units of all the time windows unit by unit is a result of the assumption that a traffic pattern will exist. This is supported by the central limit theorem, that states that when a sufficiently large number of samples is reached, the distribution of any sum or mean will approach the normal distribution. Therefore, the values for the same time unit on all windows will follow an approximately normal distribution. The Time Unit parameter has a tremendous influence on the process, because it defines the amount of data that is grouped each time. This drastically affects outlier detection process, since the detected distributions may expose different outliers depending on the used TU. Plus, it will also affect the size of the datasets retrieved for each outlier detected. Attacks can get grouped into a single TU, more isolated in each TU (ideal), or split across several TU's. This will later reflect on the retrieved datasets. Although measures were developed to identify several attacks in a single dataset, it is preferable to have the attacks the more isolated possible. This way the attack characterizations will be more accurate which will result in a better detection. When attacks are split into more than one dataset, the deviations are smaller since the attacks are split. Even so, the approach is robust and is able to still group the deviations.

This parameter can be any time measure smaller than the time window. In this experiment the used values were 1 Hour, 30 Min, 15 Min, 10 Min and 5 Min to demonstrate the detection quality variation. The maximum value of 1 Hour was chosen because the simulation data was generated on a per hour basis, therefore the pattern would be distorted if a greater unit was used (e.g. 2 Hour). On the contrary, units smaller than the Hour reveal more detailed patterns, and enable the detection of more outliers. The remianing values were chosen to demonstrate how this parameter affects the number of suspect clicks detected and the final detection quality. In the future, an optimization heuristic will be designed to automatically define the Time Unit. This is one of the essential steps for the full automatization of the process.

### 4.5.4 Modified Z-Test Confidence (ZC)

The confidence parameter is cumulative percentage used to calculate the Z-Score that will be used as a threshold for labeling a data point as an outlier or not. This "confidence" was used instead of the actual Z-Score because the word confidence and a percentage-like value is more intuitive to the user than the original parameter name and value range. As described previously in Section 3.4, for each data point its modified Z-Score will be calculated and will be compared with a threshold value. As stated before, the test heuristic states that value with a score greater than 3.5 (which corresponds to a $\sim 0.998$ confidence) should be labelled as an outlier. In this experiment, the values used for this parameter were 0.999999, 0.998 and 0.99, which correspond to the Z-Score values of $\sim 4.75$, $\sim 3.5$ and $\sim 2.3$, respectively. Higher confidence values give more certainty that the labelled deviations

are in fact outliers, which will result in higher detection accuracy and lower false positive ratio. Even so, "milder" outliers might be discarded that in the worst case could contain an attack, thus increasing the false negative ratio. It is then clear that it is important to test several values in order to find a balance between these measures. Figure 3.6 illustrates the several existing relationships in the Normal Distribution, containing the cumulative percentages and the Z scores.

### 4.5.5 Dimension Similarity Threshold (DST)

When characterizing the attacks, the several detected outliers on the different dimensions must be grouped to characterize the attack and define the attributes to be used in the clustering algorithm. The Dimension Similarity Threshold (Expression 3.2) is the threshold for the deviation between each dimension click count deviation and the maximum visit count deviation (between the estimated attack size and all dimensions). The purpose of this parameter is to detect the characteristics that completely define the attack, ignoring those that partially define it. When no characteristic is found that completely defines the attack, it is assumed that whether more than one attack is present on the outlier data or it is a false positive/too heterogeneous attack. In the experiments the used value was 0.15 to provide an accurate match while having some flexibility. At the end of this dimension grouping a new value is calculated which consists of the mean of all deviations, for a more accurate estimate of the attack size.

### 4.5.6 Cluster Size Similarity Threshold (CSST)

After clustering the outlier datasets, the task to choose the cluster that most likely corresponds to the attack emerges. To do so clusters are filtered by their size relative to the estimated size of the attack. The Cluster Size Similarity Threshold (Expression 3.4) is the threshold for the deviation between cluster size and the estimated attack's size (number of clicks). Values above this threshold will be discarded from further analysis. A complete dscription can be found in Subsection 3.7.1. Since this parameter is only to filter clusters that may contain the attack and not to actually choose the cluster, some flexibility can be given. Although unnecessary processing is not desired due to setting it too high, some margin can be given to cover more possibilities. This said, the used parameter value was 0.2 because it provides a good matching criteria without loosing flexibility and without adding too much extra processing.

### 4.5.7 Cluster Similarity Threshold (CST)

Once a cluster is marked as candidate it still has to pass one last test to be chosen. This test consists of checking if the cluster contains the expected characteristics, namely elements

Table 4.2: Experimental Configuration parameters and values.

| No. | TW | TU | ZC | DST | CSST | CST |
|---|---|---|---|---|---|---|
| 1 | 1 Day | 1 Hour | 0.999999 | 0.15 | 0.2 | 0.8 |
| 2 | | | 0.9998 | | | |
| 3 | | | 0.99 | | | |
| 4 | | 30 Min | 0.999999 | | | |
| 5 | | | 0.9998 | | | |
| 6 | | | 0.99 | | | |
| 7 | | 15 Min | 0.999999 | | | |
| 8 | | | 0.9998 | | | |
| 9 | | | 0.99 | | | |
| 10 | | 10 Min | 0.999999 | | | |
| 11 | | | 0.9998 | | | |
| 12 | | | 0.99 | | | |
| 13 | | 5 Min | 0.999999 | | | |
| 14 | | | 0.9998 | | | |
| 15 | | | 0.99 | | | |

with the attribute values that match the suspected attack ones. To do so, each data point of the dataset is checked to see if its attributes values match the attack characteristics and if so a counter is incremented. Afterwards a ratio between this counter and the estimated attack size is calculated and then compared to the parameter value that acts as a threshold. The complete explanation can be found in Subsection **??**. In this experiment, the used value was 0.8 because it requires some precision and yet has some flexibility.

## 4.6 Experimental Configuration

The seven parameters controlling the fraud detection process are shown in Table 4.2 with their different values. Each row corresponds to a parameter configuration and will generate an experiment and its resulting evaluation metrics.

Table 4.3: DBSCAN Parameters, their descriptions and used values.

| Parameter | Description | Value |
|---|---|---|
| Epsilon ($\varepsilon$) | $\varepsilon$-Neighbourhood radius | 0.2 |
| Min Points | Minimum number of points to form a cluster | 10 |

The DBSCAN algorithm was used to discover the clusters from different outlier data. Table 4.3 shows the DBSCAN parameters and their values. Changing the values of these parameters might affect the resulting cluster quality and thus detection quality. They were chosen based on previous experiments with all kinds of scenarios. Since this thesis aims to study the changes in fraud detection quality, and not the performance of DBSCAN, these parameter values will be the default values for all experiments.

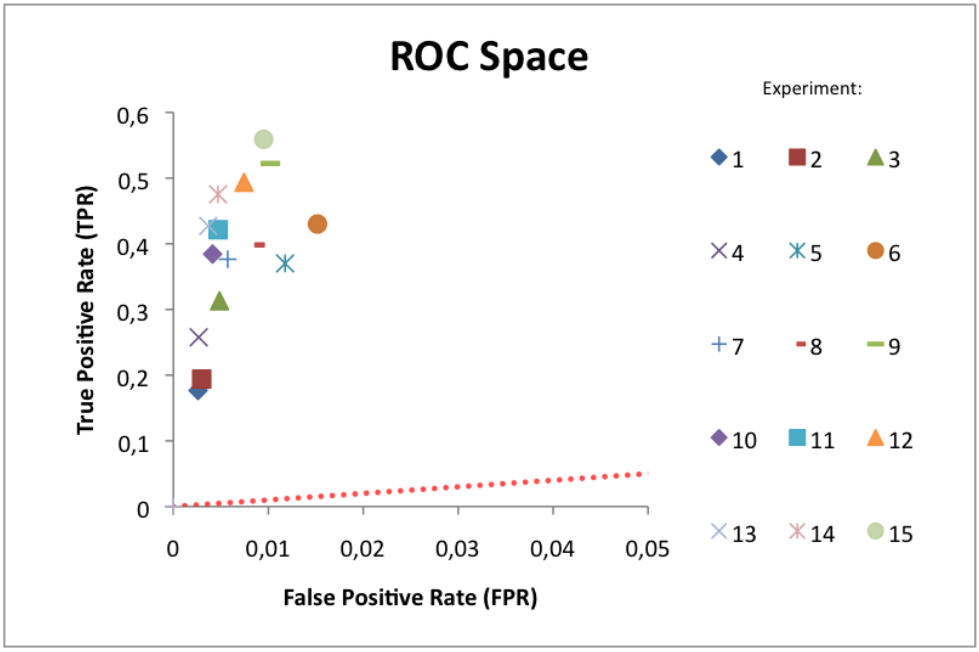


Figure 4.1: ROC chart of the conducted experiments.

## 4.7 Experimental Results

This section presents the experimental results obtained. Several different types of charts are provided to enable a good visualization of the used metrics. Each chart is analyzed and some conclusions are taken. First, a ROC chart is presented in Figure 4.1, enabling a global view of the detection performance. The accuracy (ACC) evolution is plotted in Figure 4.2, which demonstrates how the used parameters affect the detection outcome. Then, Subsections 4.7.1, 4.7.2 and 4.7.3 provide charts and analize the Detection Rates, Amount of Detected Clicks and Average Scores, respectively. The Detection Rates studied are the FPR and FNR, since they are the two error metrics and provide a good perspective of the detection performance. A complete table of the experimental results can be found in Appendix B.

Figure 4.1 illustrates the results of the experiments in the ROC Space. As mentioned in Subsection 2.2.3, these type of chart is very common when evaluating binary classification systems. Therefore, it is also adequate for Fraud Detection systems. The dotted red line represents the diagonal, whih is equivalent to a random guess detection (50-50). Results above the diagonal are considered to be good results. A perfect classification would be located at $(0, 1)$, which means no false positives and no false negatives. We can see by looking at the figure that all the experiments had a good result, since they are above the diagonal. The FPR is extremely low for all experiments, which is a very good indicator. Relatively to the TPR, it improves as the experiments advance. This indicates that more

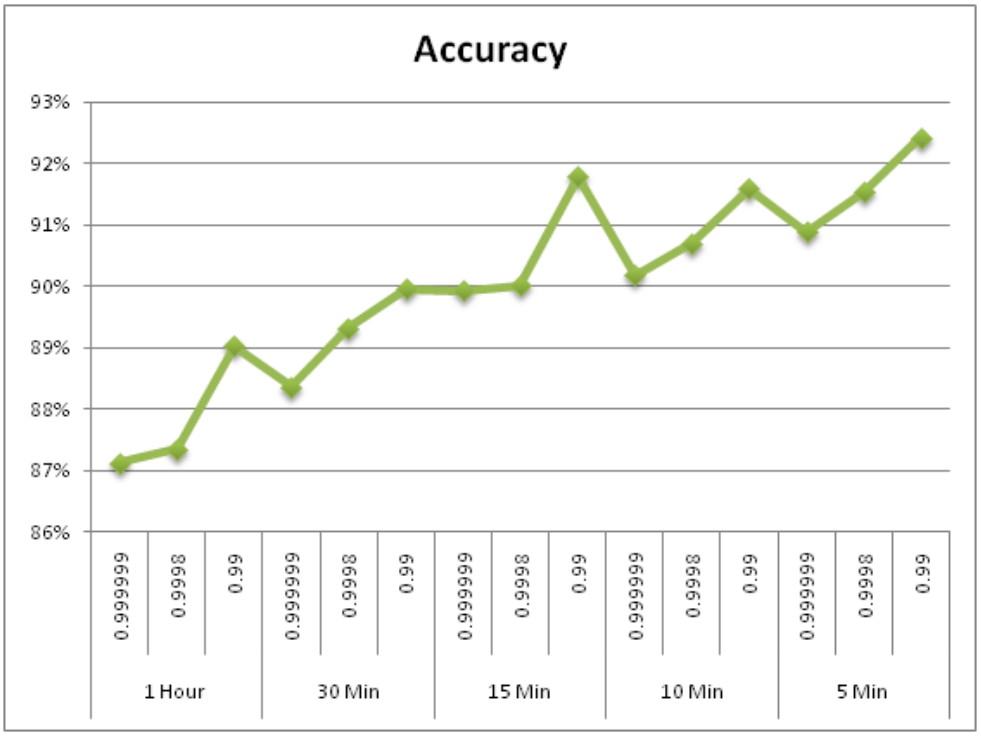


Figure 4.2: Accuracy (ACC) evolution.

invalid clicks were detected especially due to shorter TU's. As predicted before, smaller TU's reveal traffic patterns with more details. In turn, these traffic patterns reveal more outliers which lead to more attacks being detected. On the contrary, larger TU's miss these smaller variations in the traffic patterns. Obviously, many more attacks pass undetected.

Figure 4.2 provides a clear view of the accuracy evolution during the several experiments.The two main conclusions to be made are that smaller TU's tend to increase detection accuracy, as do lower ZC's. The accuracy is calculated using the correct classifications (true positives and true negatives). As mentioned above, smaller TU's generate more outliers which increase the number of invalid clicks detected (True positives). Furthermore, by isolating more the attacks less valid clicks are clustered together with the invalid clicks. This results in less false positives, which is the same as more true negatives.

### 4.7.1 Detection Rates

In this subsection, the experimental results are discussed in terms of the FPR and FNR. Since these are the two error rates, they accurately show the detection performance. Being the error rates, it is intuitive that they must be minimized. Each of the error rates is plotted into a chart, and an analysis is made.

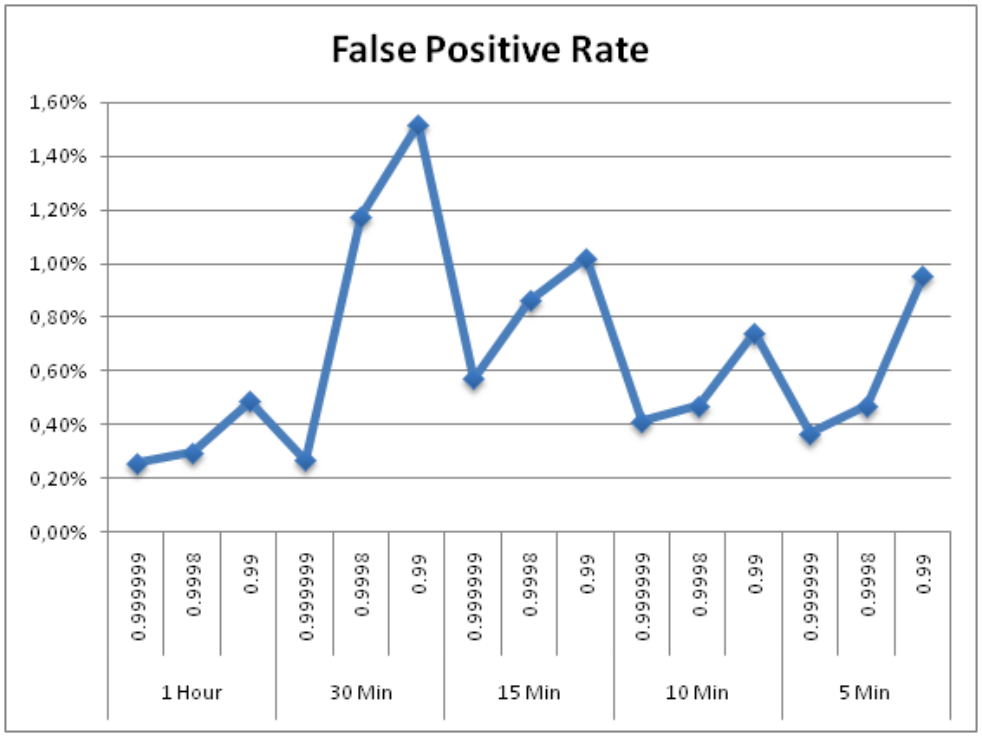


Figure 4.3: False Positive Rate (FPR) evolution.

Figure 4.3 illustrates the evolution of the FPR thru the conducted experiments. This metric is used to evaluate the detection performance in terms of the portion of valid clicks that were detected as invalid. The first observation is that during all the conducted experiments the FPR was extremely low. The maximum value was $\sim 1.5\%$, and the minimum $\sim 0.26\%$. As expected, higher ZC's reflected on a smaller number of false positives. As the ZC increased, more outliers were detected. This results in a higher probability of detecting false outliers (thus more false positives). However, it also enables the detection of "milder" outliers that can contain attacks. More attacks result in more clustering runs. Since in each clustering run some valid clicks are clustered together with the attacks, the false positives also increase. Even so, by analyzing the remaining results, this small increase in the FPR is acceptable given the increase in the number of true positives.

Figure 4.3 shows the behaviour of the FPR during the several experiments. This metric reflects the portion of invalid clicks that were not detected. It is the error that has higher costs to the advertiser, since more fraud goes by undetected. It is also the harder error to minimize, because of the vast amount of fraud techniques, their variations, and constant evolution. However, it has been stated before that the goal of this dissertation is not to solve the click fraud problem. This dissertation proposes a novel approach to detect some types of click fraud. These types ara characterized by causing deviations in normal traffic patterns. By analyzing the chart, a clear evolution can be seen along the experiments.

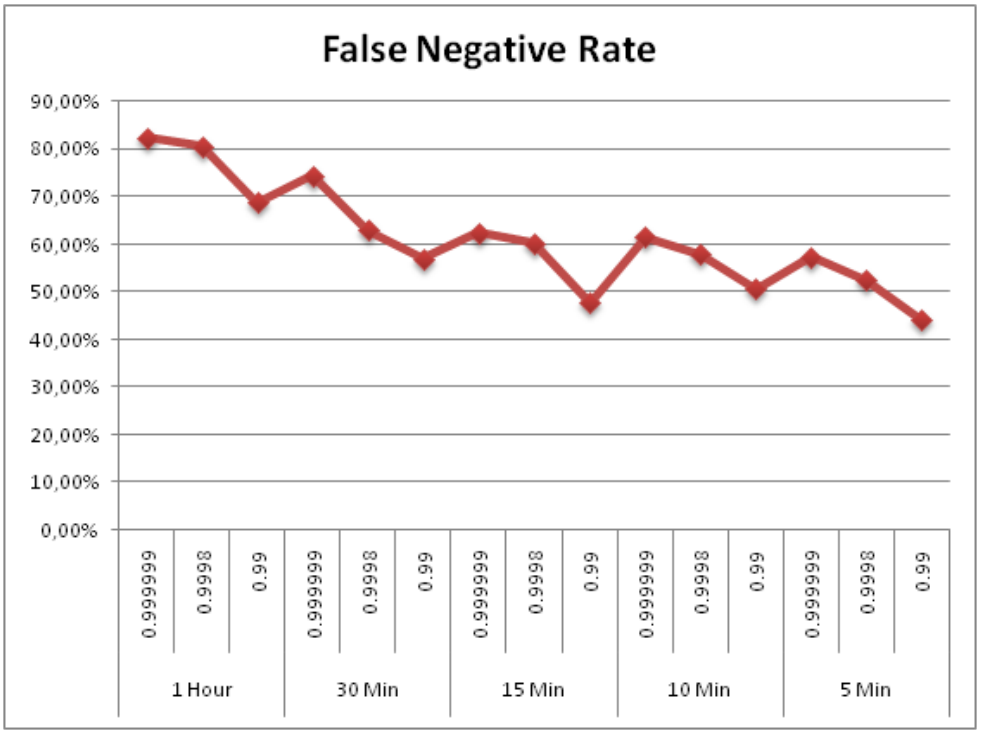


Figure 4.4: False Negative Rate (FNR) evolution.

The conclusion is once again simple: using smaller TU's and lower ZC's contributed to lower FPR's. The reason is exactly the same as mentioned before. These parameters have a great influence on the final outcome, and smaller TU's and lower ZC's increase the number of true positives, which obviously results in less false negatives. Using a 1 Hour TU the best result was a FNR of $\sim 68\%$, while when using a 5 Min TU a FNR of $\sim 44\%$ was achieved. This represents roughly 20% more clicks detected in relation to the total amount of fraud. This big difference would certainly be much appreciated by the advertiser. A 44% of undetected fraud traffic may seem quite significant, but it is already considered a very good result consideding the goal of this dissertation.

### 4.7.2 Detected Clicks

This section contains the experimental results relative to the amount of detected clicks. These are divided into accurately detected clicks (Detected Fraud) and incorrectly detected clicks (False Positives). Figure 4.5 graphically represents the results, where each bar corresponds to a experimental configuration, and clearly reveals the proportion of clicks in the final set. This results validate once more what has already been said before about the influence of the parameters.

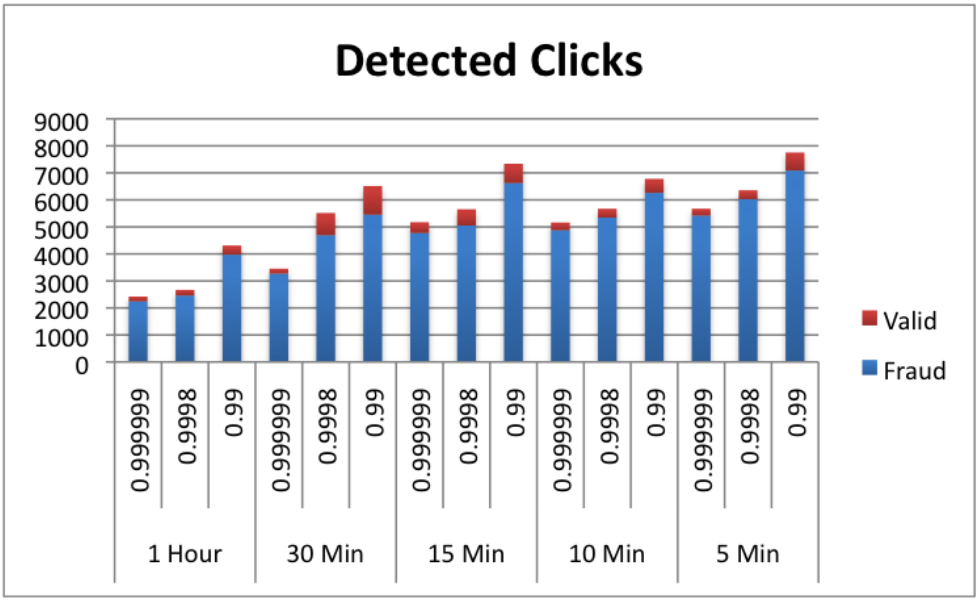


Figure 4.5: Amount and distribution of the detected clicks.

### 4.7.3 Traffic Quality Average Scores

One of the requirements of the framework was that it should be able to provide a score for each click. As described before, this score represents a measure of traffic quality and is used to minimize the false positive rate. Ultimately, the more accurate this score is, the less error the AuditService scoring engine will have. Like said before, the goal is to minimize the fraudulent clicks score while maximizing the valid clicks' score. Figure 4.6 illustrates the evolution of these metrics across the different experiments, and allows the visualization of some interesting facts. Recall that this score is based on the deviation of the cluster size relatively to the estimated attack size.

Looking at the chart the main conclusion is that the scores normally increase when the ZC decreases, which is predictable. This is even more evident when looking at the Average Valid Score, which is a good indicator of the robustness of the framework. This is due to the fact that with less confidence, although more clicks are detected also less certainty the system has that they in fact are fraud. Put in another way, the cluster sizes deviate more of the predicted attack size and thus higher scores are given. Concluding, even though the scoring formula is extremely simple it proves to perform quite well.

## 4.8 Conclusions

This chapter presented the results of applying the proposed approach on the generated dataset described before. Obviously, it would be better if other experiments could have been made, but unfortunately it was not possible during the project because of several

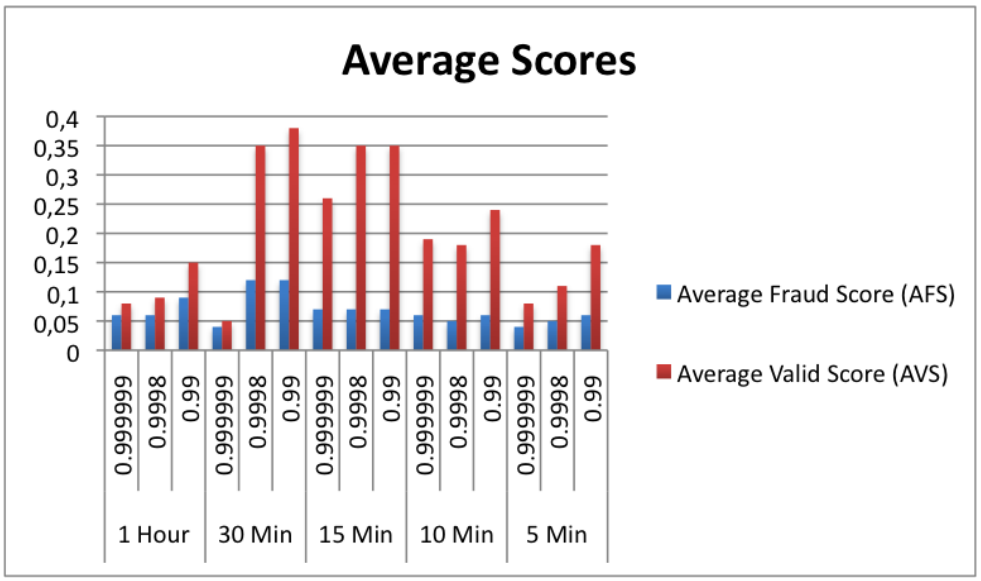


Figure 4.6: Average Scores for the detected clicks.

factors. In the future the framework will be tested using generated datasets with different traffic patterns and fraud scenarios, and ultimately with real traffic datasets.

It can be stated that the Time Unit and Z Confidence parameters have a tremendous influence on the detection performance. Therefore, special attention must be given when choosing their values. By analyzing the additional application screenshots in Appendix C, one can easily see the influence of both parameters in the statistical analysis. Figures C.1, C.2, C.3, C.4 and C.5 illustrate the influence that the TU parameter has on the amount of detected outliers. And, by comparing the Figures C.3 and C.6, one can easily observe the effect the ZC parameter has on the number of detected outliers as well.

The results of the experiments showed that using a time unit of 5 Minutes and Z Confidence of 0.99 yielded the best overall results. The detection quality kept improving by reducing the time unit as seen on Figures 4.1, 4.2 and 4.4. Using smaller TU's more attacks are detected, get more isolated and are better characterized, resulting in smaller errors. As a consequence of the previous the number of detected clicks also improved, reaching a maximum of 56% of the total fraud present on the dataset being correctly detected(Figure 4.5). Additionally, lower thresholds for the modified z-score test allow "milder" outliers to be detected. These "milder" outliers cause the detection of more attacks, improving the detection quality without compromising too much the FPR (which was always below 2%). The scores also improved using this configuration, which can be verified by looking at Figure 4.6.

The main goal of this approach is met, since this approach was able to detect all those attacks that can be detected by it. Even though only 56% of the total fraud was detected, $\sim$11.8% of the existent fraud on the generated dataset corresponded to single

person attacks. These attacks don't cause a deviation on the traffic to be detected as an outlier, given the small number of clicks. Plus, there were two botnet attacks correctly detected as outliers that due to their heterogeneity couldn't have their clicks differentiated. The botnet profile of both attacks was "Multiple Everything", where the clicks don't have any attribute in common. These two attacks, one with 1208 clicks and the other with 1106, corresponded to 18% of all fraud. The remaining $\sim$14% (44%-18%-12%) were attacks that once more didn't cause a deviation in the traffic patterns, even though they had some characteristic in common. Therefore, these 44% of undetected click fraud merely reflect the limitations of this approach. With the exception of these "undetectable" cases, this approach was able to detect all those types of click fraud attacks that caused deviations on the traffic patterns.

By analyzing these experimental results, it is considered that this methodological approach is valid. Using a set of metrics commonly used in Fraud Detection problems, it was possible to verify that this approach is accurate and robust. Even though the tests were conducted in a simulated environment, they demonstrated that the methodological steps are capable of detecting click fraud attacks that cause deviations in normal traffic patterns. Therefore, the main goal is accomplished.

# Chapter 5

# Conclusion

## 5.1 Final Remarks

From the analysis of the related work, several conclusions could be made. These conclusions essentially defined the methodological approach.

The AuditMark's AuditService analysis briefly described the auditing platform where the solution will be integrated. The architecture and the most relevant modules were described. Special attention was given to the Tag-Based Data Collection module, and more specifically to the JavaScript Interaction Code (JIC). Some of the most important attributes collected by the JIC were mentioned, but the complete list was not mentioned for compromising proprietary information. It can be said however that including the data collected thru Java, the total amount of attributes is close to 300 (although there is some attribute redundancy).

Even so, it is clear that the gathered data is extremely rich. It contains all the attributes traditional web server logs provide, as well as many more attributes that would be impossible to extract from traditional web server logs. However, there is one serious limitation, which is the fact that the retrieved information is only relative to the landing page. This seriously limits navigational pattern analysis. By knowing the pages the user visited, one could extract valuable information such as paths and page visit times.

In a Fraud Detection problem, the appropriate method to use depends on the available data's characteristics. If labelled cases of both fraudulent and legitimate cases exist, one can consider using supervised methods. Otherwise, only unsupervised methods can be employed. Considering the problem at hands, where there are no known examples of both types, the usage of unsupervised methods became clear. A review of the used metrics was also presented, with special emphasis to the classical measures resulting from the binary classifier confusion matrix. It was observed that several approaches use suspicion scores as the detection output as well. From a validation point of view, the quality of these

scores can be seen as a performance indicator. The set of metrics used in this approach were adapted from Click Fraud detection scenarios.

After analyzing the Click Fraud problem and existing approaches, one can conclude that this problem cannot be completely solved by a single approach. This is mainly due to the vast amount of existing techniques, each with its specific variations. Thus, creating a method that accurately detects all of them is virtually impossible. Fraudsters are constantly creating new click fraud techniques. Therefore every approach must be able to adapt and will be in constant improvement. Another conclusion is that approaches should be designed for a specific point of the business. This increases the probability of the approach being effectively deployed, since fewer entities are affected. In the proposed approach, only a specific type of click fraud is being targeted. The type of fraud being detected consists of those attacks that are likely to cause deviations in normal traffic patterns (such as click farms, affiliated networks and botnets). Finally, the solution will be implemented only in the advertiser's web sites. Even though less information is available, this makes the deployment easier to accomplish. The following section will present a complete review of Web Usage Mining state-of-the-art. It will allow a precise definition of the most adequate techniques to employ in the context of this click fraud detection.

The Web Usage Mining techniques that can be employed on a given problem also depend primarily on the available data's characteristics. The decision of using unsupervised techniques had already been made when studying Fraud Detection's state-of-the-art. Hence, after analyzing the available data and the problem at hands, the chosen techniques were EDA and cluster analysis. EDA provides extremely valuable information on the data. In this context, it models the base traffic distributions and detects deviations from it. These deviations allow differentiating some clicks that are suspect of being invalid. To differentiate these groups of clicks that are potentially invalid, the choice of using cluster analysis became clear. Additionally, given the nature of the click fraud problem, marking a given click as invalid is very context-dependant, and extremely difficult. Hence, no labels (valid or invalid) can be given to the clicks. This leads to unlabelled datasets, which automatically exclude supervised techniques such as classification and regression. As a consequence of the data collection method being used, only the clicks performed on the landing pages (pages where the ads point to) had their information collected. Thus, no sessions could be extracted because the collected clicks all belonged to the same page. So, techniques that explore user session data were automatically discarded. These are Sequential and Navigational patterns analysis techniques. In this context, Association and Correlation analysis techniques have the potential of describing the traffic's characteristics in terms of rules. This could aid in characterizing the traffic patterns, to then detect deviations. Even though, they were not applied and were left for future developments.

Relatively to the proposed methodology, it was designed for detecting only some types of click fraud attacks. These attacks can be described as attacks that have characteristics

that are likely to cause deviations on the base distributions of the underlying valid traffic. These deviations are detected by the use of statistical methods. Then the deviations are analyzed, which allow the differentiation of the clicks that effectively caused each deviation. Once the clicks are detected, they are given a suspicion score. This score is based on the existing degree of confidence that those clicks in fact contain invalid traffic.

Since this approach is based primarily on a statistical outlier analysis, the base distribution (traffic patterns) will differ from web site to web site. This distribution and consequent patterns may become visible on one website by setting the analysis time window to 1 day, while on another they become visible only when using a 1 week time window, and so on. It may even happen that the amount of collected traffic may not be enough for any traffic patterns to emerge, in which case this approach will not be effective. When no traffic pattern emerges there are only two possible ways considering the current approach. The first is to try and find the analysis time window that reveals the inherent traffic pattern of the website. If no time window reveals the traffic pattern it is possibly due to not having collected enough data, in which case the analysis should be postponed to a moment where more data is available. If both of these solutions fail, it is possible that the analyzed web site possesses a chaotic traffic pattern, which is impossible to analyze using this approach. Even so, this is unlikely considering the central limit theorem. So it can be stated that with enough data patterns will emerge under the form of normal distributions. Nevertheless, the correct analysis time window must still be used.

The proposed approach is considered innovative and robust. It defines a clear workflow for the detection process of some types of click fraud. It is important to state that this approach is a proof-of-concept that demonstrates the applicability of web usage mining techniques for the click fraud detection problem. The theoretical steps were described in detail for a generic scenario, but the experiments were conducted under simpler scenarios. Some extensions and adaptations may have to be made to prepare this approach for a real life scenario.

The results of the experiments showed that the chosen values for the time division process and statistical outlier parameters are the most critical. Using smaller time intervals for performing the statistical analysis results in smaller errors. More attacks are detected, they get more isolated and are better characterized. As a consequence of the previous the number of detected clicks also increases. Using lower thresholds for the statistical outlier detection (modified z-score) also provided better results. The reason was that “milder” outliers were detected, which in turn contained attacks previously undetected. The detection quality kept improving by varying these parameters as can be seen on Figures 4.1, 4.2 and 4.4. The best result was 56% of the total fraud present on the dataset being correctly detected(Figure 4.5). The scores also improved when using shorter time intervals and lower thresholds for the modified z-score test, which can be verified by looking at Figure 4.6. Even though, all of the experimental results are considered good when analyzing

the ROC chart in Figure 4.1.

By analyzing these experimental results, it is considered that this methodological approach is valid. Using a set of metrics commonly used in Fraud Detection problems, it was possible to verify that this approach is accurate and robust. Even though the tests were conducted in a simulated environment, they demonstrated that the methodological steps are capable of detecting click fraud attacks that cause deviations in normal traffic patterns. Therefore, the main goal is accomplished. Web Usage Mining techniques were succesfully employed to the click fraud detection problem, and proved to be extremely useful in this context. A functional prototype was developed, which serves as proof-of-concept. However, only a small portion of the full potential of Web Usage Mining was explored, and much more work can be done in this specific area.

## 5.2 Future Work

Some further developments include exploring other techniques of Web Usage Mining. Association and Correlation analysis can be very helpful in discovering valuable rules in the traffic data. Rules can be found for characterizing both the base traffic distribution, and for detecting outliers. Furthermore, they can be used for analyzing the outlier data in search of rules that better define the possible attacks. If information about the user's sessions starts to be collected, it will enable the usage of Sequential and Navigational analysis. This family of techniques has the potential of discovering a completely new set of traffic patterns. Traffic patterns relative to the paths commonly followed by the users, page visit's durations and others would tremendously enrich a click fraud detection approach. Obviously, these patterns could be used exploring the methodology described in this approach.

Relatively to the current approach, several developments can and will be made. The traffic simulator can be extended to use more characteristics for defining both the valid and invalid clicks. The click fraud attacks and their profiles can also be improved to create more realistic attacks. The time division process for the statistical analysis will be automated, using some sort of heuristic using optimization techniques. A similar approach will be used for determining the optimal parameter for the outlier detection. This will enable a fully automated detection process. The statistical analysis, outlier detection and clustering analysis can and will be modified to use a larger number of attributes. The scoring formula can be refined to differentiate even better the detected clicks.

In terms of implementation, the solution can be improved in many ways. The number of database queries can be reduced, and the queries optimized. The application can be parallelized, to be able to analyze larger amounts of data, faster. The several modules can be separated to form a distributed system, to distribute the load amongst several machines.

Furthermore, this approach can be adapted to other scenarios with different purposes. In the context of a store, analyzing the data relative to it's clients could reveal hidden trends in shopping habits. It could also detect sudden increases or decreases of visitors, bought products, money spent, etc. The clients that defined these deviations could be characterized for future study. For instance, this could result in valuable information about the response a given marketing campaign was getting. A similar adaptation could be made for instance in an hospital. The information about the incoming new pacients, could reveal good/bad responses to medication or even the outbreak of diseases. Analyzing the rate of pacients, their symptoms and diagnosis could aid in discovering periodic patterns, which could aid in medication stock planning.

These same principles of modelling the base distribution to then detect outliers can be applied in many specific domains of application. The Web Usage Mining would probably be "plain" Data Mining in some cases, but the same principles would apply.

# Appendix A

# Created Attack Profiles for the Simulation

Table A.1: Sets of profiles created for each click fraud attack type.

| **Attack Type** | **Profile Name** | **Prob.** | **NClicks** | **Dur.** | **SingleOS** | **SingleBr** | **SingleCo** | **SingleIP** | **SingleRef** |
|---|---|---|---|---|---|---|---|---|---|
| Single Person | Single Everything | 0.5 | 7 | 20 | T | T | T | T | T |
| | Change Browser | 0.1 | 10 | 30 | T | F | T | T | T |
| | Change IP | 0.2 | 5 | 20 | T | T | T | F | T |
| | Change Country | 0.2 | 7 | 20 | T | T | F | F | T |
| Click Farm | Multiple IP's and Referrers | 0.4 | 100 | 5 | T | T | T | F | F |
| | Multiple Browsers and IP's | 0.3 | 150 | 10 | T | F | T | F | T |
| | Single Everything | 0.1 | 50 | 2 | T | T | T | T | T |
| | Single Country and Referrer | 0.2 | 200 | 20 | F | F | T | F | T |
| Affiliated Click Fraud | Single Country | 0.6 | 200 | 20 | F | F | T | F | F |
| | Multiple Country and Single Referrer | 0.4 | 150 | 10 | F | F | F | F | T |
| Botnet | Single Browser and Referrer | 0.15 | 1000 | 5 | T | T | F | F | T |
| | Single OS and Referrer | 0.15 | 1000 | 5 | T | F | F | F | T |
| | Single OS | 0.15 | 1000 | 10 | T | F | F | F | F |
| | Single Browser and Referrer | 0.1 | 900 | 7 | F | T | F | F | T |
| | Single Browser | 0.15 | 500 | 10 | F | T | F | F | F |
| | Single Referrer | 0.1 | 800 | 8 | F | F | F | F | T |
| | Multiple Everything | 0.1 | 1200 | 4 | F | F | F | F | F |
| | Ghost Botnet | 0.1 | 1200 | 60 | F | F | F | F | F |

Legend:

Prob.: Probability
NClicks: Number of Clicks
Dur.: Attack Duration (minutes)
SingleBr: Single Browser
SingleCo: Single Country
SingleRef: Single Referrer
F: False
T: True

# Appendix B

# Experimental Results Table

Table B.1: Experimental results table containing all the used metrics.

| TW | TU | ZC | Detected | TP | FN | FP | TN | ACC | TPR | FNR | FPR | TNR | AFS | AVS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 Day | 1 Hour | 0.999999 | 2421 | 2240 | 10443 | 181 | 69609 | 87.12% | 17.66% | 82.34% | 0.26% | 99.74% | 0.06 | 0.08 |
| | | 0.9998 | 2668 | 2460 | 10223 | 208 | 69582 | 87.35% | 19.40% | 80.60% | 0.30% | 99.70% | 0.06 | 0.09 |
| | | 0.99 | 4131 | 3972 | 8711 | 339 | 69451 | 89.03% | 31.32% | 68.68% | 0.49% | 99.51% | 0.09 | 0.15 |
| | 30 Min | 0.999999 | 3453 | 3267 | 9416 | 186 | 69604 | 88.36% | 25.76% | 74.24% | 0.27% | 99.73% | 0.04 | 0.05 |
| | | 0.9998 | 5514 | 4693 | 7990 | 821 | 68969 | 89.32% | 37.00% | 63.00% | 1.18% | 98.82% | 0.12 | 0.35 |
| | | 0.99 | 6513 | 5454 | 7229 | 1059 | 68731 | 89.95% | 43.00% | 57.00% | 1.52% | 98.48% | 0.12 | 0.38 |
| | 15 Min | 0.999999 | 5173 | 4774 | 7909 | 399 | 69391 | 89.93% | 37.64% | 62.36% | 0.57% | 99.43% | 0.07 | 0.26 |
| | | 0.9998 | 5652 | 5051 | 7632 | 601 | 69189 | 90.02% | 39.82% | 60.18% | 0.86% | 99.14% | 0.07 | 0.35 |
| | | 0.99 | 7337 | 6624 | 6059 | 713 | 69077 | 91.79% | 52.23% | 47.77% | 1.02% | 98.98% | 0.07 | 0.35 |
| | 10 Min | 0.999999 | 5162 | 4874 | 7809 | 288 | 69502 | 90.18% | 38.43% | 61.57% | 0.41% | 99.59% | 0.06 | 0.19 |
| | | 0.9998 | 5673 | 5344 | 7339 | 329 | 69461 | 90.70% | 42.14% | 57.86% | 0.47% | 99.53% | 0.05 | 0.18 |
| | | 0.99 | 6779 | 6260 | 6423 | 519 | 69271 | 91.58% | 49.36% | 50.64% | 0.74% | 99.26% | 0.06 | 0.24 |
| | 5 Min | 0.999999 | 5674 | 5418 | 7265 | 256 | 69534 | 90.88% | 42.72% | 57.28% | 0.37% | 99.63% | 0.04 | 0.08 |
| | | 0.9998 | 6356 | 6028 | 6655 | 328 | 69462 | 91.53% | 47.53% | 52.47% | 0.47% | 99.53% | 0.05 | 0.11 |
| | | 0.99 | 7753 | 7089 | 5594 | 664 | 69126 | 92.41% | 55.89% | 44.11% | 0.95% | 99.05% | 0.06 | 0.18 |

| | |
|---|---|
| Total Clicks | 82473 |
| Total Fraud Clicks | 12683 |
| Total Valid Clicks | 69790 |

# Appendix C

# Additional Application Screenshots

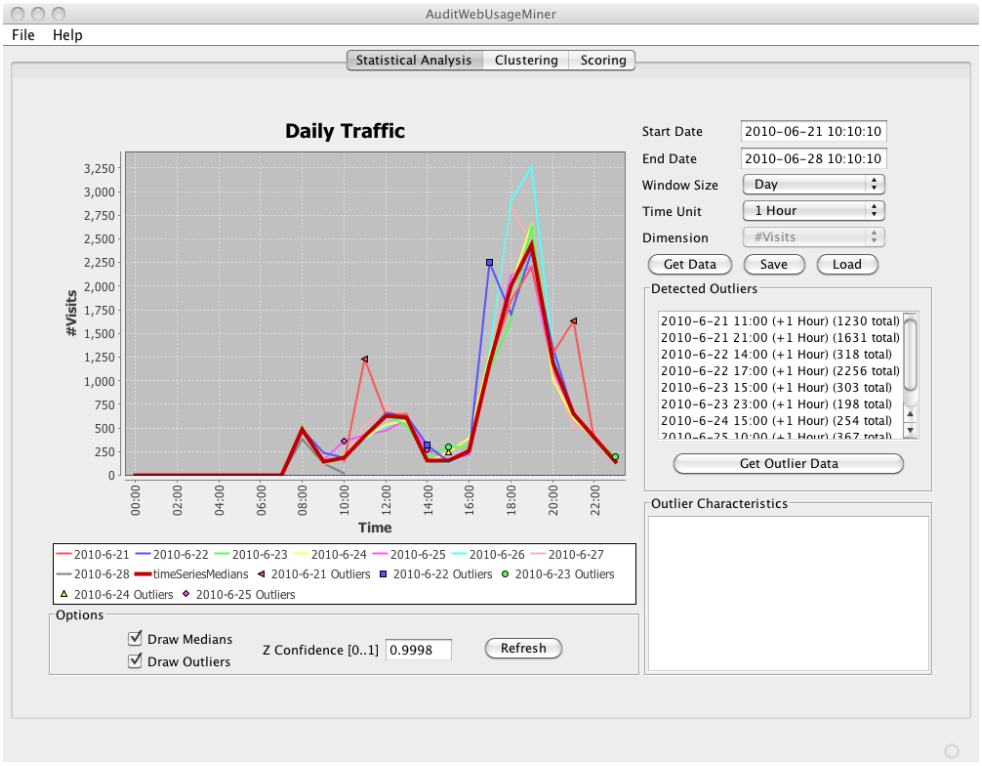


Figure C.1: AuditWebUsageMiner Statistical Module (TU = 1 Hour, ZC = 0.9998)

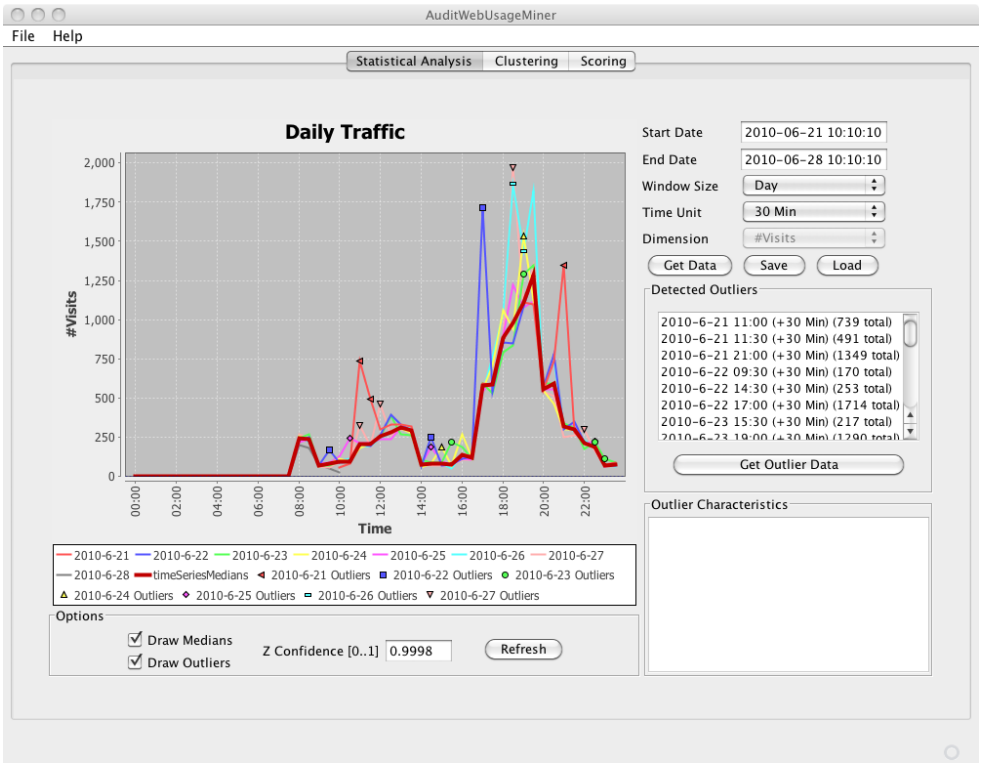


Figure C.2: AuditWebUsageMiner Statistical Module (TU = 30 Min, ZC = 0.9998)

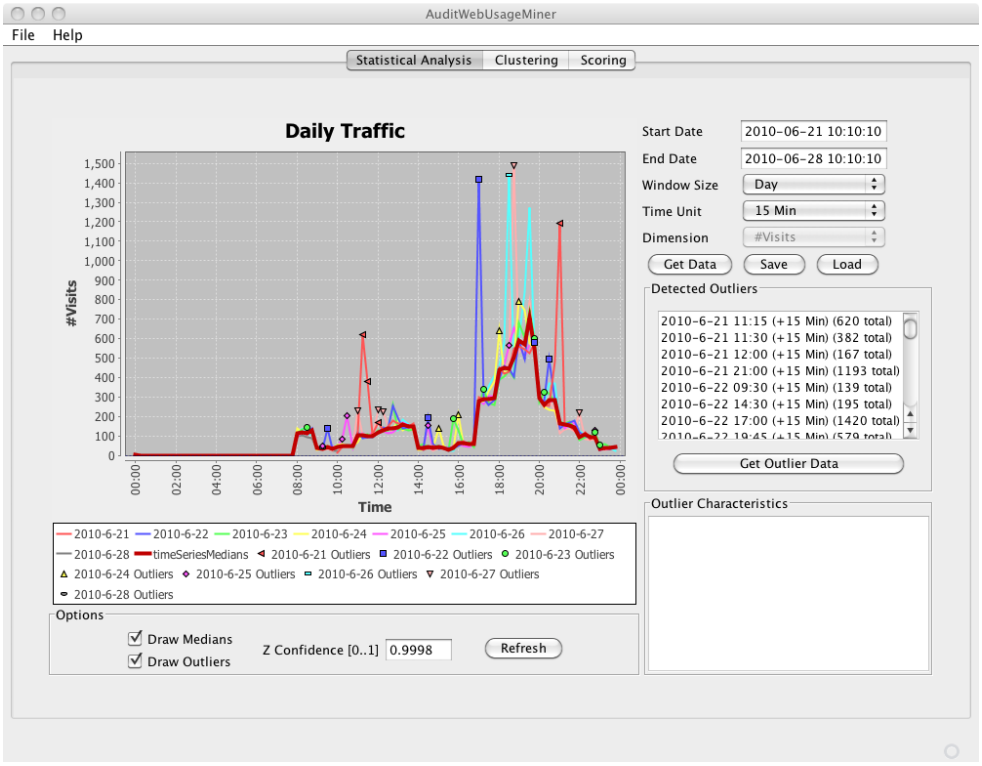


Figure C.3: AuditWebUsageMiner Statistical Module (TU = 15 Min, ZC = 0.9998)

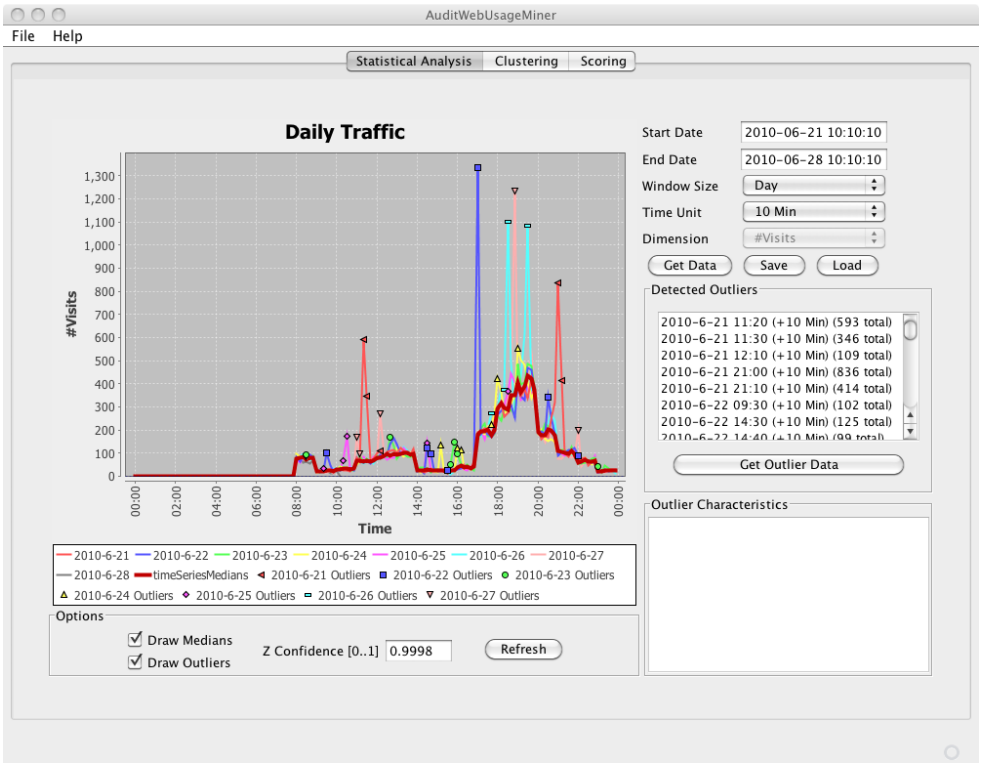


Figure C.4: AuditWebUsageMiner Statistical Module (TU = 10 Min, ZC = 0.9998)

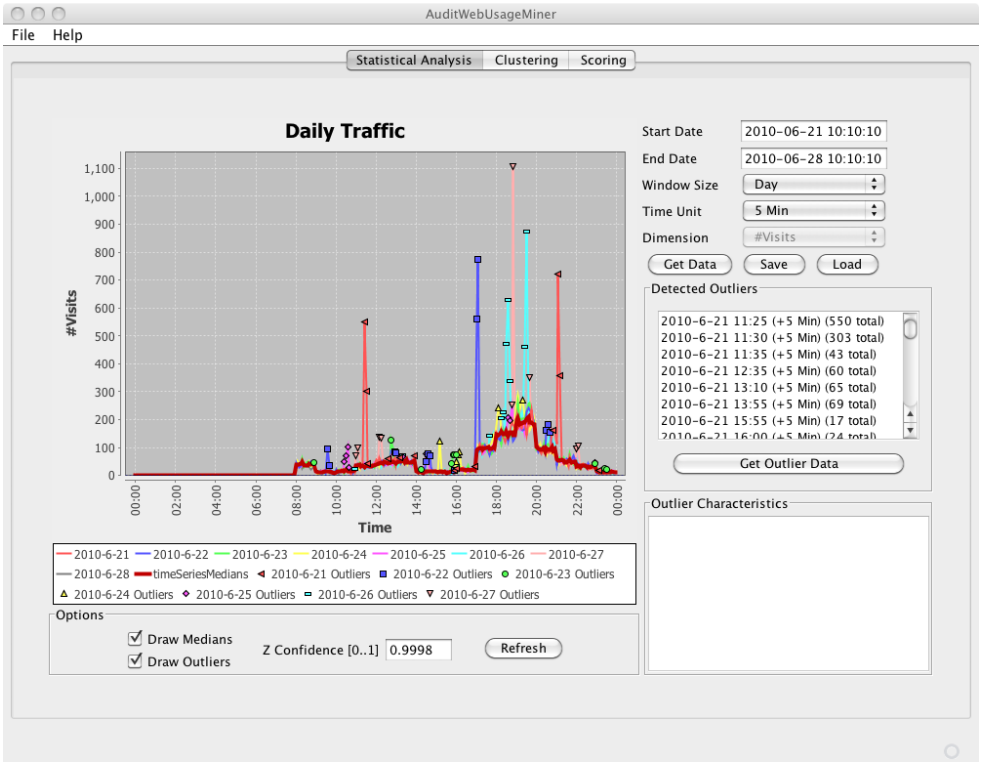


Figure C.5: AuditWebUsageMiner Statistical Module (TU = 5 Min, ZC = 0.9998)

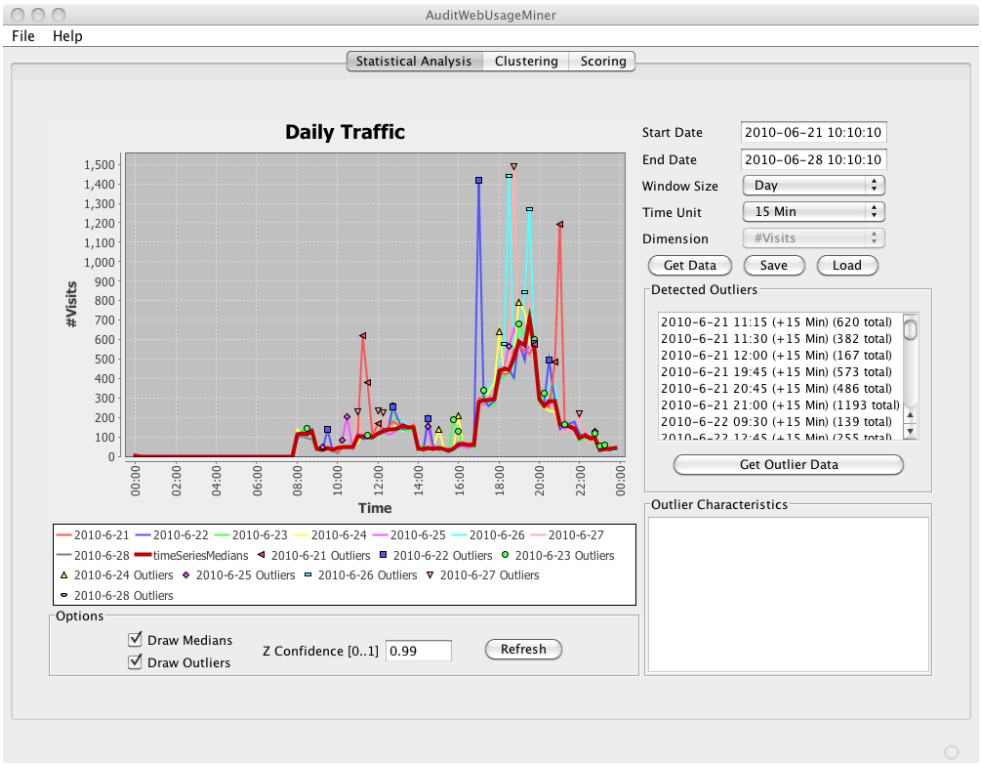


Figure C.6: AuditWebUsageMiner Statistical Module (TU = 15 Min, ZC = 0.99)

# References


[Aho09] Ahorre.com. 2009 internet advertising revenue. *http://www.ahorre.com/dinero/internet/web/2009_internet_advertising_revenue/ (last visited on 28/06/2010)*, 2009.

[Anc10] AnchorIntelligence.com. Traffic quality report. *http://www.anchorintelligence.com/ai/resources/category/traffic_quality_report/, last visited on 28th June 2010*, 2010.

[BBD+02] Patrick L. Brockett, Patrick L. Brockett, Richard A. Derrig, Linda L. Golden, Arnold Levine, and Mark" Alpert. Fraud classification using principal component analysis of ridits. *Journal of Risk and Insurance*, 69(3):341–371, 2002.

[BBHH01] Richard J. Bolton, Richard J. Bolton, David J. Hand, and David J." H. Unsupervised profiling methods for fraud detection. *PROC. CREDIT SCORING AND CREDIT CONTROL VII*, pages 5–7, 2001.

[BD93] Iglewicz B. and Hoaglin D.C. How to detect and handle outliers. *ASQC basic references in quality control*, 16, 1993.

[Ben00] Peter J. Bentley. "evolutionary, my dear watson" - investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims. In L. Darrell Whitley, David E. Goldberg, Erick Cantú-Paz, Lee Spector, Ian C. Parmee, and Hans-Georg Beyer, editors, *GECCO*, pages 701–709. Morgan Kaufmann, 2000.

[BGMP99] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedreschi. A classification-based methodology for planning audit strategies in fraud detection. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 175–184, New York, NY, USA, 1999. ACM.

[BHH02] Richard J. Bolton, David J. Hand, and David J. H. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–255, 2002.

[BKJ03] E.L. Barse, H. Kvarnstrom, and E. Jonsson. Synthesizing test data for fraud detection systems. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 384 – 394, 8-12 2003.

[BL84] V. Barnett and T. Lewis. *Outliers in statistical data*. Wiley and Sons, 1984.

[BL00] José Borges and Mark Levene. Data mining of user navigation patterns. *Web Usage Analysis and User Profiling*, 1836:92–112, 2000.

[BLH99] R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 103 –106, 1999.

[BM98] Alex G. Büchner and Maurice D. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Rec.*, 27(4):54–61, 1998.

[BS00] Bettina Berendt and Myra Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal*, 9(1):56–75, March 2000.

[BST01] Peter Burge and John Shawe-Taylor. An unsupervised neural network approach to profiling the behavior of mobile phone users for use in fraud detection. *Journal of Parallel and Distributed Computing*, 61(7):915 – 925, 2001.

[CCHC04] Rong-Chang Chen, Ming-Li Chiu, Ya-Li Huang, and Lin-Ti Chen. Detecting credit card fraud by using questionnaire-responded transaction model based on support vector machines, 2004.

[CFPS99] Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, 14(6):67–74, 1999.

[CHM+03] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, October 2003.

[CKK02] Yoon Ho Cho, Jae Kyeong Kim, and Soung Hie Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329 – 342, 2002.

[CKR98] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. *SIGCOMM Comput. Commun. Rev.*, 28(4):241–253, 1998.

[CL07] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 10–10, Berkeley, CA, USA, 2007. USENIX Association.

[Cli09] ClickForensics.com. Industry click fraud rate at 15.3 percent for q4 2009. *http://www.clickforensics.com/newsroom/press-releases/158-industry-click-fraud-rate-at-153-percent-for-q4-2009.html, last visited on 28th June 2010*, 2009.

[CLPS02] Michael H. Cahill, Diane Lambert, José C. Pinheiro, and Don X. Sun. Detecting fraud in the real world. *Handbook of massive data sets*, pages 911–929, 2002.

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[CNM04] Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78, New York, NY, USA, 2004. ACM.

[CP01] Corinna Cortes and Daryl Pregibon. Signature-based methods for data streams. *Data Mining and Knowledge Discovery*, 5(3):167–182, July 2001.

[CPV01] Corinna Cortes, Daryl Pregibon, and Chris Volinsky. Communities of interest, 2001.

[CT04] Chuang-Cheng Chiu and Chieh-Yuan Tsai. A web services-based collaborative scheme for credit card fraud detection. In *e-Technology, e-Commerce and e-Service, 2004. EEE '04. 2004 IEEE International Conference on*, pages 177 – 181, 28-31 2004.

[DDLR77] A. P. Dempster, A. P. Dempster, N. M. Laird, and D. B." Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[DGSC97] J.R. Dorronsoro, F. Ginel, C. Sgnchez, and C.S. Cruz. Neural fraud detection in credit card operations. *Neural Networks, IEEE Transactions on*, 8(4):827 –834, jul 1997.

[DK04] Mukund Deshpande and George Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Technol.*, 4(2):163–184, 2004.

[DMR+08] Neil Daswani, Chris Mysen, Vinay Rao, Stephen Weis, Kourosh Gharachorloo, Shuman Ghosemajumder, and the Google Ad Traffic Quality Team. *Crimeware: Understanding New Attacks and Defenses*, volume Online Advertising Fraud. Addison-Wesley Professional, 2008.

[DSTT07] Neil Daswani, Michael Stoppelman, Google Quality Team, and Google Security Team. The anatomy of clickbot.a. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[DT97] A. Deshmukh and T.L.N. Talluru. A rule based fuzzy reasoning system for assessing the risk of management fraud. In *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, volume 1, pages 669 –673 vol.1, 12-15 1997.

[E.95] Cox. E. A fuzzy system for detecting anomalous behaviors in healthcare provider claims. *Intelligent Systems for Finance and Business*, pages 111–134, 1995.

[EKJX96] Martin Ester, Hans-peter Kriegel, S. Jörg, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise, 1996.

[Fan04] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 128–137, New York, NY, USA, 2004. ACM.

[FBH00] Xiaobin Fu, Jay Budzik, and Kristian J. Hammond. Mining navigation history for recommendation. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 106–112, New York, NY, USA, 2000. ACM.

[FP97] Tom Fawcett, 1, Foster Provost, and 1. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, September 1997.

[FP99] Tom Fawcett and Foster Provost. Activity monitoring: noticing interesting changes in behavior. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 53–62, New York, NY, USA, 1999. ACM.

[FS01] Dean P. Foster and Robert A. Stine. Variable selection in data mining: Building a predictive model for bankruptcy. Center for Financial Institutions Working Papers 01-05, Wharton School Center for Financial Institutions, University of Pennsylvania, February 2001.

[GGJR06] Mona Gandhi, Mona Gandhi, Markus Jakobsson, and Jacob" Ratkiewicz. Badvertisements: Stealthy click-fraud with unwitting accessories. *ONLINE FRAUD, PART I JOURNAL OF DIGITAL FORENSIC PRACTICE, VOLUME 1, SPECIAL ISSUE 2*, 1:2006, 2006.

[Gho06] Shuman Ghosemajumder. Let click fraud happen? uh, no. *http://googleblog.blogspot.com/2006/07/let-click-fraud-happen-uh-no.html (last visited on 28/06/2010)*, 2006.

[GKM99] John Garofalakis, Panagiotis Kappos, and Dimitris Mourloukos. Web site optimization using page popularity. *IEEE Internet Computing*, 3(4):22–29, 1999.

[GR94] S. Ghosh and D.L. Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Vol.III: Information Systems: Decision Support and Knowledge-Based Systems, Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621 –630, 4-7 1994.

[Han81] D. J. Hand. *Discrimination and Classification*. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. John Wiley & Sons Inc, 1981.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[Hof01] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, pages 177–196, 2001.

[HT98] Jaakko Hollmén and Volker Tresp. Call-based fraud detection in mobile communication networks using a hierarchical regime-switching model. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 889–895, Cambridge, MA, USA, 1998. MIT Press.

[HWGH97] Hongxing He, Jincheng Wang, Warwick Graco, and Simon Hawkins. Application of neural networks to detection of medical fraud. *Expert Systems with Applications*, 13(4):329 – 336, 1997. Selected Papers from the PACES/SPICIS'97 Conference.

[JM02] D.R. Riedinger J.A. Major. Efd: A hybrid knowledge/statistical-based system for the detection of fraud. In *Journal of Risk & Insurance*, volume 69, pages 309–324, 2002.

[JSJ07] Ari Juels, Sid Stamm, and Markus Jakobsson. Combating click fraud via premium clicks. In *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–10, Berkeley, CA, USA, 2007. USENIX Association.

[JZM05] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. A maximum entropy web recommendation system: combining collaborative and content features. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 612–617, New York, NY, USA, 2005. ACM.

[KB00] Raymond Kosala and Hendrik Blockeel. Web mining research: a survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, 2000.

[KOO03] Jungwon Kim, A. Ong, and R.E. Overill. Design of an artificial immune system as a novel anomaly detector for combating financial fraud in the retail sector. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 1, pages 405 – 412 Vol.1, 8-12 2003.

[KTP+09] Carmelo Kintana, David Turner, Jia-Yu Pan, Ahmed Metwally, Neil Daswani, Erika Chin, and Andrew Bortz. The goals and challenges of click fraud penetration testing systems. *International Symposium on Software Reliability Engineering*, 2009.

[KWW+08] M. Kantardzic, C. Walgampaya, B. Wenerstrom, O. Lozitskiy, S. Higgins, and D. King. Improving click fraud detection by real time data fusion. In *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on*, pages 69 –74, 16-19 2008.

[LA09] Naeimeh Laleh and Mohammad Abdollahi Azgomi. A taxonomy of frauds and fraud detection techniques, 2009.

[Lar05] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley-Interscience, 2005.

[LHM99] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 337–341, New York, NY, USA, 1999. ACM.

[LX01] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, pages 130 –143, 2001.

[McL92] G. J. McLachlan. *Discriminant analysis and statistical pattern recognition*. Wiley, New York, 1992.

[MCS00] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[MDLN01] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *WIDM '01: Proceedings of the 3rd international workshop on Web information and data management*, pages 9–15, New York, NY, USA, 2001. ACM.

[MDLN02] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Min. Knowl. Discov.*, 6(1):61–82, 2002.

[MdR94] Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20(Supplement 1):629 – 679, 1994.

[MEH+99] Yves Moreau, Lerouge Ellen, Verrelst Herman, Stormann Christof, and Burge Peter. A hybrid system for fraud detection in mobile communication. *EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS*, pages 447–454, 1999.

[MKR07] S. Majumdar, D. Kulkarni, and C.V. Ravishankar. Addressing click fraud in content delivery systems. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 240 –248, 6-12 2007.

[ML07] Zdravko Markov and Daniel T. Larose. *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*. Wiley-Interscience, 2007.

[Mob07] Bamshad Mobasher. Data mining for web personalization. *The adaptive web: methods and strategies of web personalization*, pages 90–135, 2007.

[MTVM02] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick. Credit card fraud detection using bayesian and neural networks. In *Proceedings of the First International NAISO Congress on Neuro Fuzzy Technologies*, 2002.

[MWGM08] Bob Mungamuru, Stephen Weis, and Hector Garcia-Molina. Should ad networks bother fighting click fraud? (yes, they should.). Technical Report 2008-24, Stanford InfoLab, July 2008.

[PAL04] Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *SIGKDD Explor. Newsl.*, 6(1):50–59, 2004.

[PB98] R.A. Derrig P.L. Brockett, X. Xia. Using kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. *Journal of Risk and Insurance*, 65(2):245–274, 1998.

[PP99] James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USITS'99: Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems*, pages 13–13, Berkeley, CA, USA, 1999. USENIX Association.

[PPKS02] G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. D. Spyropoulos. Discovering user communities on the internet using unsupervised machine learning techniques. *Interacting with Computers*, 14(6):761 – 791, 2002.

[PPPM03] Foster Provost, Foster Provost, Claudia Perlich, and Sofus A." Macskassy. Relational learning problems and simple models. *IJCAI 2003 WORKSHOP ON LEARNING STATISTICAL MODELS FROM RELATIONAL DATA (SRL-2003*, pages 116–120, 2003.

[PPPS03] Dimitrios Pierrakos, Georgios Paliouras, Christos Papatheodorou, and Constantine D. Spyropoulos. Web usage mining as a tool for personalization: A survey. *User Modeling and User-Adapted Interaction*, 13(4):311–372, November 2003.

[PVS05] Jagdish Pathak, Navneet Vidyarthi, and Scott Summers. A fuzzy-based algorithm for auditors to detect elements of fraud in settled insurance claims. *Managerial Auditing Journal*, 20(6):632–644, 2005.

[PZCG10] Yanlin Peng, Linfeng Zhang, J. Morris Chang, and Yong Guan. An effective method for combating malicious scripts clickbots. *Computer Security – ESORICS 2009*, 5789:523–538, 2010.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[RMN+99] Saharon Rosset, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. Discovery of fraud rules for telecommunications—challenges and solutions. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 409–413, New York, NY, USA, 1999. ACM.

[Sar00] Ramesh R. Sarukkai. Link prediction and path analysis using markov chains. *Computer Networks*, 33(1-6):377 – 386, 2000.

[SF99] Myra Spiliopoulou and Lukas C. Faulstich. Wum: A tool for web utilization analysis. *The World Wide Web and Databases*, 1590:184–203, 1999.

[SG01] B. Stefano and F. Gisella. Insurance fraud evaluation: a fuzzy expert system. In *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, volume 3, pages 1491 –1494, 2001.

[SGJ07] Travis Schluessler, Stephen Goglin, and Erik Johnson. Is a bot at the controls?: Detecting input data attacks. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 1–6, New York, NY, USA, 2007. ACM.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[SKS98] Stuart Schechter, Murali Krishnan, and Michael D. Smith. Using path profiles to predict http requests. *Computer Networks and ISDN Systems*, 30(1-7):457 – 467, 1998. Proceedings of the Seventh International World Wide Web Conference.

[SN03] Kate A. Smith and Alan Ng. Web page clustering using a self-organizing map of user navigation patterns. *Decision Support Systems*, 35(2):245 – 256, 2003.

[Spi00] Myra Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134, 2000.

[SZC02] Hua Shao, Hong Zhao, and Gui-Ran Chang. Applying data mining to detect fraud behavior in customs declaration. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 3, pages 1241 – 1244 vol.3, 2002.

[SZP02] M. Syeda, Yan-Qing Zhang, and Yi Pan. Parallel granular neural networks for fast credit card fraud detection. In *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, volume 1, pages 572 –577, 2002.

[THHT98] M. Taniguchi, M. Haft, J. Hollmen, and V. Tresp. Fraud detection in communication networks using neural and probabilistic methods. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1241 –1244 vol.2, 12-15 1998.

[Tuz06] Alexander Tuzhilin. The lane's gifts v. google report. Technical report, New York University, 2006.

[vA97] Constantin von Altrock. *Fuzzy logic and NeuroFuzzy applications in business and finance*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[VDD04] S. Viaene, R.A. Derrig, and G. Dedene. A case study of applying boosting naive bayes to claim fraud diagnosis. *Knowledge and Data Engineering, IEEE Transactions on*, 16(5):612 – 620, may 2004.

[w3s10] w3schools.com. Os and web browser statistics. *http://www.w3schools.com/browsers/browsers_os.asp and http://www.w3schools.com/browsers/browsers_stats.asp, last visited on 28th June 2010*, 2010.

[WA00] R. Wheeler and S. Aitken. Multiple algorithms for fraud detection. *Knowledge-Based Systems*, 13(2-3):93 – 99, 2000.

[WFYH03] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, New York, NY, USA, 2003. ACM.

[WH97] Graham J. Williams and Zhexue Huang. Mining the knowledge mine: The hot spots methodology for mining large real world databases. In *AI '97: Proceedings of the 10th Australian Joint Conference on Artificial Intelligence*, pages 340–348, London, UK, 1997. Springer-Verlag.

[Wil99] Graham J. Williams. Evolutionary hot spots data mining - an architecture for exploring for interesting discoveries. In *PAKDD '99: Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining*, pages 184–193, London, UK, 1999. Springer-Verlag.

[WZ09] Kenneth C. Wilbur and Yi Zhu. Click fraud. *Marketing Science*, 28(2):293–308, 2009.

[YH03] Alexander Ypma and Tom Heskes. Automatic categorization of web pages and user clustering with mixtures of hidden markov models. *WEBKDD 2002 - MiningWeb Data for Discovering Usage Patterns and Profiles*, 2703:35–49, 2003.

[YiTWM04] Kenji Yamanishi, Jun ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, May 2004.

[ZD02] Dell Zhang and Yisheng Dong. A novel web usage mining approach for search engines. *Computer Networks*, 39(3):303 – 310, 2002.

[ZG08] Linfeng Zhang and Yong Guan. Detecting click fraud in pay-per-click streams of online advertising networks. In *Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference on*, pages 77 –84, 17-20 2008.