University of Kent

A Machine Learning Approach To Detecting Insider Threat In Access Control System

Thesis Presented To

The University Of Kent At Canterbury

In The Subject Of Computer Science

For The Degree of MSc. Networks and Security

Sara Al Kindi. SMAA7

Course Title: Project Research

Course Code: CO880

Date: 15th September 2014

Word Count: 13200

Acknowledgment

Words may fail a person in expressing gratitude and appreciation to those who earnestly deserve it. First of all I would like to thank the Ministry of Higher Education in Oman, the Cultural Attaché of Oman in UK, and Petroleum Development of Oman (PDO) for allowing me the chance to pursue my studies and accomplish my goals abroad, and for assisting me financially and academically throughout my studies.

My thanks extends to my supervisor Dr Rogerio de Lemos, who guided me throughout this project and helped me to reach the aimed goals needed from me till the thesis came to completion.

In addition, I would like to thank and say my gratitude to the PhD student Chris bailey. He helped me put the grounds of the study, went through the technical process of my project, and gave me useful comments and beneficial remarks.

Also, I would like to express my thanks to all my wonderful friends whom I came across and was honoured to share beautiful memories here in Kent. Asdaf, Khadeejah, Abeer, Majda, May, Maha, Rerma and Nawal. Thank you girls for everything, you were my home and safe shelter during stressful periods I went through this year.

Last but not least, my profound thanks and sincere words of appreciation are due to my family. My mother and father for their continues prayers and support in every choice we make. My brother Fahad thank you for your help and support whenever I am down and gloomy. My thanks extend to my husband, Hamad, for his love and support during this year abroad.

Thank you.

Contents

List of figures

List of tables

Abstract

Machine learning tools are effective mechanisms to handling insider threat within access control systems. It is a complementary approach to promising solutions that handle insider threat through self-adaptation. Self-adaptation promotes the automated run-time adaptation of a system, for example, an access control system, to adapt to system and environment change , such as, malicious behaviour. A problem facing self-adaptive systems (in the context of handling insider threat), is the challenge to ensure accurate identification of both known and unknown attacks. The Naïve Bayes algorithm and WEKA tool have been selected to demonstrate the working principle for this approach. This project illustrates the insider threat problem with the aid of run time gamification in order to trace both malicious behavior and corresponding self-adaptive response simultaneously. The effectiveness of this approach to capture anomalies in user behavior has been investigated through four main phases: I. classification using game application data to analyse detected abuses at the resource level, II. classification using authorization level data to analyse the detected abuses at the system level, III. classification using the combination of both application and authorization data to extend detection scope, and IV. fusing game and authorization classifiers results to refine overall classification decision. Resource and system levels malicious behavior features were found complementing each other leading to a wider scope of anomalies detection using both levels' context simultaneously. The use of this tool along with self-adaptive systems will provide a better control over suspicious behavior within authorisation infrastructure.

Chapter 1

Introduction

One of the most organizational security concerns that grasped a huge worldwide attention is tackling insider threats problem due to the importance to recognize any insiders attempt to violate organization policies and misuse their access privileges to admit illegitimate behavior **(Hunker, J., & Probst, C. W., 2011)**. Consequently, organisations have to identify carefully the appropriate tools that are capable of monitoring, controlling and detecting anomalies behavior to keep track of logs that could capture possible attacking attempts from intruders.

There are a wide range of intrusion detection techniques that vary in their detection scope , such as, capabilities and target applications which will be introduced and briefed in details at literature review chapter. Moreover, Intrusion detection techniques are commonly classified into two main classes which are misuse detection and anomaly detection. Misuse detection techniques are capable to identify known attack patterns , such as, signature based detection methods **(Bace, R., & Mell, P., 2001)**. Whereas, anomaly detection approaches focus on detecting anomalies throughout activity patterns **(Eskin, E., et al, 2002)**. Currently, intrusion detection community are expanding their detection scope where they are looking for more advanced approaches , such as, machine learning techniques **(Mukkamala, S., et al, 2002).**

However, machine learning concept is based on automatic learning of normal processes, activities and normal range of interactions. Then, learning tools will be capable of analysing behavior patterns and looking for anomalies and any variance in normal activities, which will give an indication of possible attacks that are rarely captured by traditional detection systems. Hence, machine learning analysis tools would be able to identify the new fingerprints of attackers malicious behavior in the provided set of data , such as, servers logs, access logs, user profiles and authorizations (**Dodson, S., 2014).** This entire process of applying computer intelligence techniques to acquire new knowledge from data is called data mining (**Kantardzic, M. , 2011).** Learning can be tested by observing changes in behavior

and comparing it with past recorded behavior where the learning can be converted from knowledge base to performance base **(Witten, Ian H. et al, 2011).**

1.1 Problem statement

The significance of threats posed from insiders could come with much higher risk rate than outsiders as a result of their accompanied privileges to access organization resources. Hence, some detection techniques focused heavily on access control approaches to control authorization and to organize the resources. For example, Self-Adaptive Authorization Framework (SAAF) is designed to manage the policy which is based on authorization infrastructure of distributed Role Based Access Control authorization model (RBAC) and Attribute Based Access Control authorization model (ABAC), by assigning relevant permissions. It is intended to monitor the usage of authorization infrastructure autonomously whenever an abnormal behavior is detected and execute the right decision "on whether to adapt the authorization infrastructure or not". The main challenge of self-adaptive systems is to ensure the accurate identification of known and unknown attacks **(Bailey, C. et al, 2014)**. Similarly, other researchers proposed other approach that focus on enhancing intrusion detection systems on access control system **(Bertino, E., et al, 2005) (Shin, D., & Claycomb, W. R., 2010)**. However, most of the cases, detection patterns could be identified for the detector , such as, working with a predefined set of rolls and attributes which narrow detection scopes to specify attacks only.

Likewise, there are some identified limitations in the preceding literature approaches to detect insider threats in access control systems if their patterns are not determined. Those limitations initiated the necessity of more advanced detection techniques that are capable to look outside the defined circumstances of attacks and learn from activities to capture unknown threats in access control systems.

1.2 Project Objective

This project aims is to apply a simple machine learning algorithm to detect anomalies in user behavior, which can be used as an input for self-adaptive access control system. In addition, there are certain objectives need to be achieved which are as follows:

i. It intends to investigate the performance of machine learning approaches to capture anomalies in user behavior and analyse detected abuses at the resource level.

ii. The effectiveness of anomaly-based detection will be evaluated through machine learning experimental analysis.

1.3 Working principle

In order to conduct this project, a supervised learning algorithm which called Naïve Bayes, has been selected to implement machine learning. It is a statistical based classifier that assumes underlying probabilistic model. This algorithm will be compared with other algorithms, because it allows capturing model uncertainty through the determination of output probabilities. A real time data has been provided from other project called SAAF, as briefed above, which used an ethical hacking game data to evaluate the implementation of its Self-Adaptive Authorization Framework **(Bailey, C. et al, 2014).** The experiments in this research have been divided into four main phases, which are as follows:

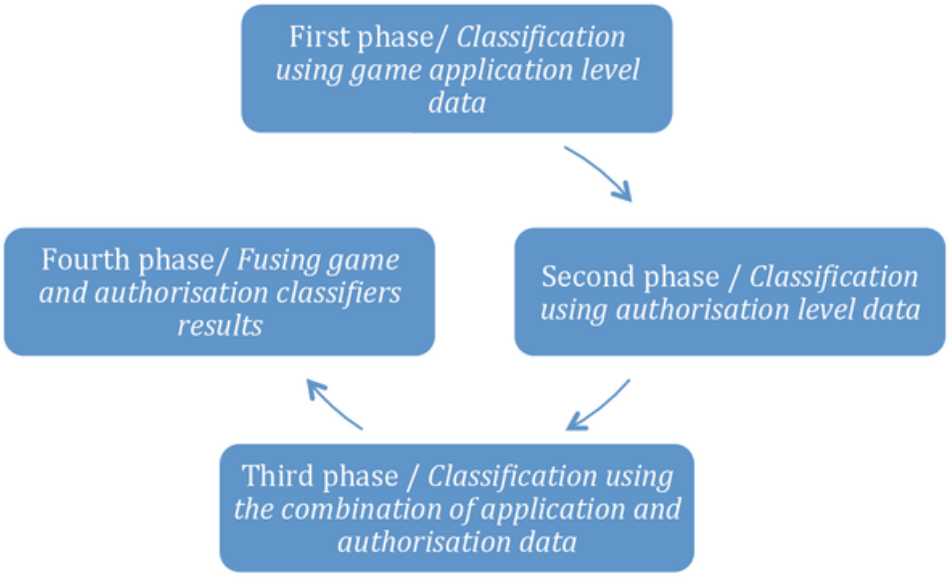


Figure 1.1 The experimental phases in this research

i. Classification using game application level data: The first experimental phase starting from extracting some application features from the game

where player violate its rules. Then, those types of features will be injected into the selected, WEKA, machine learning tool as attributes of Naïve Bayes classifier where will be used to train the classifier to distinguish between abuse and non-abuse behavior. Features (forms of attacks) will be evaluated according to the tool built in quality measures to analyse the performance of the classifier at resource level.

ii. Second phase / Classification using authorisation level data: in second phase, a new set of features will extracted at the authorization level and will be classified using Naïve Bayes algorithm as discussed in first phase.

iii. Third phase / Classification using the combination of application and authorisation data: In the third phase, both application level features and authorization level features will be combined as one data set and injected in machine-learning tool. Then it will be classified and evaluated accordingly to observe the integration effect on classifier performance.

iv. Fourth phase/ Fusing game and authorisation classifiers results: In the last phase, the output of first two independent classifiers will be fused together using a simple OR gate to investigate the two classifiers influence on the final classification decision of user behavior.

1.4 Outline of thesis

This thesis including __ chapters and it is organized as follows:

Chapter two reveals the literature review about the problem overview as surveyed in related literature and some other work conducted to solve this problem using different intrusion detection techniques. Moreover, it briefly describes some of machine learning approaches.

Chapter three introduces the problem domain stating from overviewing authorisation infrastructures and current challenge faced by insider threats in section. Followed by, SAAF solution and its current limitations. Then, it discusses simulating insider attacks through a game environment and studies internal attacks detection throughout the game.

Chapter four evaluates some machine-learning algorithms and justifies the reasons behind Naïve Bayes selection. It also evaluates some existing tools available for Naïve Bayes implementation and gives a justification about the applicability of WEKA tool to perform experimental classification and analysis. Finally, it discusses the application methodology of suggested solution on the game.

Chapter five contains the four phases experimental results and analysis.

Chapter six includes Conclusions and future of work.

Chapter 2

Literature Review and Related Work

2.1 Insider attacks: definitions and consequences

The Vormetric Insider Threat Report at 2013 reports "Insider threats continue to present a challenge". This report reveals the insider threats that are posing an increasing risk where it becomes more difficult for organizations to detect or prevent their systems form such threats **(Oltsik, J., 2013)**. The most recent statistics surveyed by Mathew, S. et al includes several well-known IT organisations , such as, the US Secret Service, CERT (Computer Emergency Response Team), and Microsoft E-Crime report indicates that the insider attacks form 34% of all surveyed attacks whereas outsiders constitute 37%, and the remaining 29% of the surveyed attacks are contributed by some unidentified causes (**Mathew, S. et al, 2010)**. A comprehensive definition of insider threat is given by CERT, where they define the malicious insider threat to an organization as "a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems" **(CERT, 2001)**. The CERT definition is highly supported by *Stolfo et al.* who characterize an insider to be usually a trusted employee or a person with authorised access rights, privileges and a higher level of trust than an outsider **(Stolfo et al, 2008)**. Historical cases of insider attacks were thoroughly analysed by CERT to which they identified five classifications of insider threat which are intellectual property (IP) theft, IT sabotage, fraud, espionage, and accidental insider threats **(CERT, 2001)**.

A great focus should be given to insider activities as they can result some risks, regardless of the intent, insignificant, direct loss in company sales, revenue, intellectual property, and reputation (**Mills, R. et al, 2009).** This may lead to an

eventual downgrading of customer confidence as revealed by *McCormac and his colleagues* who emphasized that insider activities have extended consequences that threats have to be carefully prevented, detected, and mitigated by organizations. Sasaki argued in his paper that, this could not be an easy task as insiders like employees have legitimate rights and privileges to access organization's resources to conduct their day to day work as per their roles and responsibilities **(Sasaki, T., 2012).**

1.2 Insider threats mitigation mechanisms

The rise of insider threat is becoming a substantial concern in the way of organizations operate, monitor and manage their resources. A variety of sophisticated systems and techniques have been designed to help organizations to detect, mitigate, and prevent against any insider attacks. One agreed mechanism that relies on detecting the up-normal intrusion attempts is which is called Intrusion Detection Systems. Anderson, who introduced the concept of intrusion detection, defines an intrusion attempt as "the potential possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable" **(Sundaram, A., 1996)**. The powerful contribution behind this technology is to make system capable in identifying intrusion attempts, so an appropriate corrective and preventive action can be evoked. However, as revealed by Axelsson, Stefan, and David Sands in their book, intrusion detection systems are susceptible to some problems and challenges , such as, the challenge to defend against "a hit-and-run attack" where attacker is able to predict the suspicious level of detection systems so that whenever the attacker reaches the detection threshold, it stops attacking and sends legitimate logs and reports **(Noon, E., & Li, H. , 2010).** In addition, there is a possibility of getting false positive and negative alarms, or fail to act upon getting an alarm when an attacker surreptitiously attempts to gain or succeed to gain an access **(Axelsson, S., & Sands, D., 2006)**.

Robin Sommer and *Vern Paxson* claimed that intrusion detection community experience some challenges that make it fundamentally distinct from other approaches. They argued that challenges around the intrusion detection could be originated from the outlier detection and the high costs of detection errors in terms of

finding the opposed form of the normal behavior. In addition, it is challenging to interpret the accuracy of the results phase (What does that anomaly mean?) and to evaluate the detector performance (How do we make sure it actually works?). They clarify that intrusion detection systems will require applying some tools "borrowed from machine learning community" like training data to guide the behavior of the system (What do we train our system with?) and evasion risk (Can the attacker mislead our system?). They have emphasised that, it is relatively important to answer all previous questions in order to implement the intrusion systems effectively **(Sommer, R. & Paxson, V., 2010).**

Mathew and his colleagues highly recommend the statistical approach where they build up this approach in such a way that it profiles normal users behaviors statistically based on come computing method that raise a flag when any routine deviation is detected. An effective statistical profiler would be able to identify some come insider threats , such as, "non-stealthy sabotage attacks, quick data harvesting attacks, or masquerading attacks, because of the computing footprints of those actions" is considerably different from daily activities, as can be observed from their statistical logs (**Mathew, S., et al, 2010**). In their research paper, they have modelled a data-centric approach for insider attack detection in a database system. They emphasized on the importance to specify two important things in order to construct an effective detection system which are as follows

i. The user profile: the user profile is really important thing to consider especially how it will be formed and what should be contains.
ii. The machine learning: "which machine-learning techniques and models should be adapted to specify if the profiles are practically useful for the problem detection".

Their research is parallel to other existing researches that contributed to the development of intrusion detection systems that targets databases protections from attackers. Their research has a high value due to their efforts in analysing users' interactions with an RDBMS (Relational Database Management System) by applying multiple database queries to extract relevant features and they referred this approach to 'Data Centric Approach' (**Mathew, S., et al, 2010**). It operates by constructing users profiles in the light of their working scope. Those profiles compose pairs of features and values extracted by running syntactical analysis's quires that represent their activities while working on the system. An attack would be detected by

comparing the 'fingerprints' of legitimate actions, extracted at syntactical level analysis, with user behavioral actions (**Lee, S. Y., et al, 2002).** Figure 2.1 below illustrates how the concept of profiling the user behaviors after computing the database query results and sending rolls and detection information to a centric database, which are retrieved for learning and anomalies detection phases.

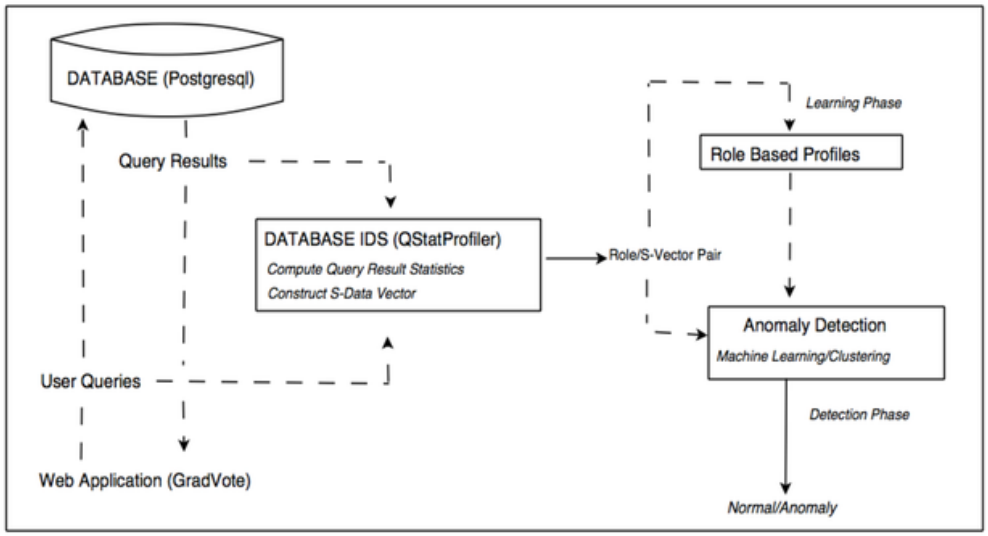


Figure 2.1 Database design for detecting insider attacks against database

In this context, *Lane, T., And Brodley* applied the same concept in their application where they build user profiles based on a similarity measure that used in command sequences. Then, it compares input sequences to the builded user profile to detect anomalies (**Lane, T., & Brodley, C. E. , 1997).**

1.3 State of the Art detections methods

Based on the current literature that have been focused on detection systems and specifically designed for insider attack detection, there are two main categories of detection approaches, surveyed in this literature review, which are:

i. Detection techniques: it incorporates social factors , such as, the psychological approach
   - Psychological Approach, Sasaki has developed "a framework that detects suspicious insiders using a psychological trigger that impels malicious insiders to behave suspiciously". His architecture composed of an announcer, a monitor, and an analyzer that is capable of triggering anomalous activities of malicious insiders , such as, data

deletion, running unwanted applications and private web browsing. This approach is recognized to be effective in preventing "information leakage via e-mail" and "illegitimate purchases". **(Sasaki, T., 2012).** Likewise, Greitzer et al. have proposed a predictive risk model that uses some psychological indicators like "disgruntlement, accepting feedback and anger management which then performs risk calculations using Bayesian network of the indicators **(Greitzer, F. L et al, 2008)**. Such approach is highly dependent on user activities and limited to psychological triggers which might not be effective in detecting all types of insider threats where behavioral activity might not be directly measured.

ii. Detection techniques: it incorporates technical factors , such as, anomaly detection approach.

- Anomaly Detection Approach, as summarized in Sasaki research paper **(Sasaki, T., 2012)**, there are various implementations of anomaly detection approach , such as, Eberle et al. who modeled the normal workflows as a graph based approach for insider threat detection. **(Eberle, W., & Holder, L.,2009).** In addition, Brancik et al. proposed a data intensive architecture comprising "event and anomaly collection", "data analysis and correlation", and "e-discovery tools" for detection and prevention from insider threats **(Brancik, K., & Ghinita, G., 2011).** Bertino et al. have proposed anomaly detection architecture for a database management system which operates by parsing and extracting some features using SQL queries so that they can be analyzed later on feature values basis **(Bertino, E., & Ghinita, G., 2011)**. Similar techniques are focused on improving the anomaly detection efficiency, which is also one of the major concerns of this research that based heavily on anomalous detection.

1.4 Insider threats in access control systems

Organizations are maintaining their access control measures and systems cautiously in order to monitor and control their user's access into organization systems and resources in their daily routine work. Furthermore, Access control

detection approaches come with rules that are defined and enforced in advance to make these approaches effective for routine work. *Claycomb and his colleague* have proposed A Malicious Activities Detection Engine (MADE) that focuses on a directory service and monitors directory's changes. Then, it produces alerts for administrators if any directory change detected by violating predefined policies **(Shin, D., & Claycomb, W. R., 2010)**. However, this approach requires that rules to be applied in advance and this might not be effective to non-routine work**. (Sandhu, R. S. et al, 1996).**

Based on access control intrusion detection systems methodology, the current and promising solution for the existing issue is self-adaptive systems that have the ability to adapt, manage, repair and update themselves automatically at run-time. Self-adaptive systems are capable of identifying internal attacks and responding to those attacks to mitigate and reduce damage. However, ensuring accurate identification of known and unknown attacks could be an issue with self-adaptive systems. *Bailey, et al* have proposed an example of such systems called a Self-Adaptive Authorization Framework (SAAF). SAAF is "capable of managing any policy based distributed authorization infrastructure" , such as, Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). SAAF implements a feedback control loop to monitor target authorization infrastructure decisions and then analyse them to judge the behavior. Finally corresponding decisions will be made after adapting the target authorization infrastructure. However, SAAF has some limitations , such as, adaptations accuracy which mostly reliant on known patterns of malicious behavior. It mostly faces a challenge with other types of malicious activities where patterns of attacks are changing under different circumstance **(Bailey, C. et al, 2011).**

Similarly, Elisa Bertino et al worked in improving the security of access control system as they recognised well that detection of "anomalous behavior by applications and users" is an important component of intrusion detection systems **(Bertino, E., et al, 2005).** They have proposed a learning solution to preform data mining on database traces by forming some user profiles which will be used to form normal patterns model so that anomalies is identified. Their approach is coupled with Role Based Access Control (RBAC) where the permissions of the system are associated with roles. Hence, insider attackers could be determined whenever they behave differently from the normal behavior of their role. As a result, it leads to some

limitation in the necessity of recoding the role information and users access patterns log files which will pose a challenge for detection system to detect unknown threats.

The result identified in the above detection systems highlight the importance of machine learning approaches to capture anomalies in user behavior and eliminate the risk of unknown attacks in complex behavior.

1.4 Machine learning approaches for anomalous detection

Most detection systems could fall under machine learning umbrella where artificial intelligence is used to construct systems that can learn from data. *Kevin Murphy* has discussed several learning algorithms that can be customized to accomplish various detection activities , such as, supervised learning algorithms, unsupervised learning algorithms, semi-supervised learning, transduction, reinforcement learning and developmental learning. **(Murphy, K. P., 2012)**.

Tsai and his colleagues have conducted a review on many machine-learning techniques by expressing the most relevant and appropriate application that can be used for. For instance:

- Pattern classifications which can used to extract predefined patterns and solve pattern recognition issues by taking raw data and activity on data category **(Michalski, R. B., & Kubat, I. M, 1998).** It is normally utilized the supervised learning by exercising training data that contain a class label and compare it with unknown data to detect if pattern matched by "computing the approximate distance between the input–output examples".
- Single classifiers which is a form of intrusion detection techniques that relies on one single machine-learning algorithm. There are various machine learning discussed in literature , such as, k-nearest neighbor **(Manocha, S., & Girolami, M. A., 2007)**, support vector machines **(Vapnik, V. N., & Vapnik, V., 1998)**, artificial neural network **(Towell, G. G., & Shavlik, J. W., 1994)**, decision trees **(Bishop, C. M., 2006)**, Naïve bayes networks and Genetic algorithms. Each individual one can be customized to tackle intrusion threats **(Tsai, C. F., et al, 2009)**.
- Ensemble classifiers, which is a multiple weak learning algorithms that are combined to improve the classification performance of a single classifier **(Kittler, J., et al, 1998)**.

Nevertheless, *what is the most effective approach that could suit this research requirement and perform the machine learning effectively in order to detect insider threats based on anomalous activities?* This research paper aims to answer all of the raised questions throughout this literature review where it will focus on implementing a simple machine learning algorithm to detect anomalies in user behavior which can be used as an input of the self-adaptive access control system. Moreover, the effectiveness of anomaly based detection through machine learning can be evaluated and provides an indication of its applicability to the self-adaptive security systems that aims.

Chapter 3

Problem Domain

3.1 Introduction

The rise of insider threat is becoming a substantial factor in the way of the organisations operate and manage their access control systems. One of the promising solutions of identifying and responding to insider threat is via the introduction of self-adaptive systems, which are capable of identifying internal attacks and responding to them by mitigating and minimizing their risk. However, those solutions are faced several limitations. However, this research illustrates the insider threat problem with the aid of gamification **(Bailey C., 2013)**. The classic board game Snakes and Ladders is a web based application, whereby the game itself is an analogy for an organisational resource. Users are invited to play the game, exploit glitches, vulnerabilities, and perform web based attacks in order to win. This type of activity is seen as synonymous with insider threat. This chapter describes this game and provides a brief overview about the data gained from this game and how insider attacks might occur at both game resources and authorization levels.

This chapter starts by giving a brief summary about authorisation infrastructures and current challenge faced by insider threats in section 3.2. Followed by, SAAF solution and its current limitations in section 3.3. Section 3.4 is simulating insider attacks through a game environment and section 3.5 discusses the internal attacks detection throughout the game. Finally the last section summarises the whole problem domain chapter.

3.2 Authorisation Infrastructures

Authorisation infrastructures govern access to the set of organisations resources , such as, PERMIS authorisation infrastructure **(Chadwick, D., et al, 2008)**, and grant users with privileges assigned to them to access organization resources, after a successful authentication. Authorisation infrastructure usually composed of four main components which are identity service, authorization service, protected resources and users. The process cycle within authorisation infrastructure usually starts by users request to authenticate their identity with identity services. After a successful

authentication, users will request the access to the organizational protected resources where request will be escalated to authorisation services as access decisions request. The later will validate the identity through identity services first to ensure that only legitimate users can gain the access. Then, authorisation services will enforce the access decisions accordingly. Figure 3.1 (**Bailey, C., 2013)** illustrates the process and the interactions between the components.

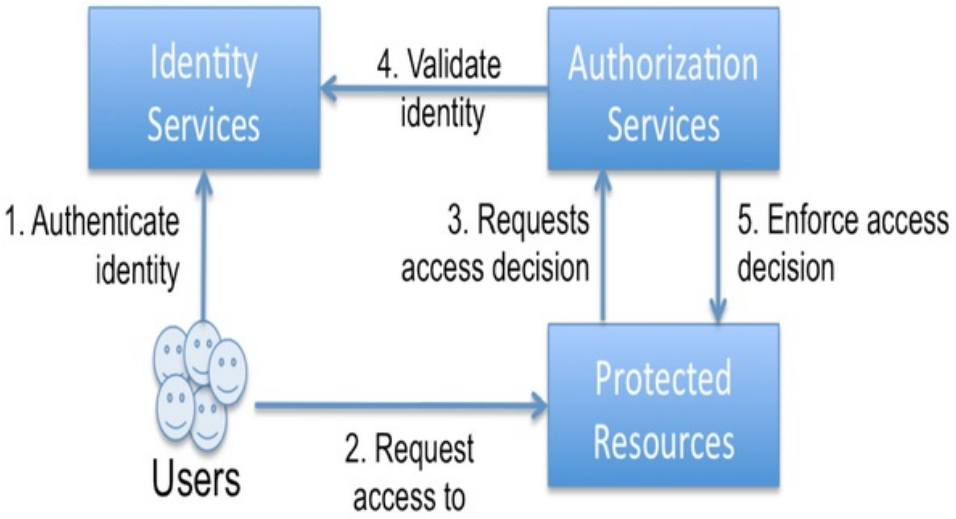


Figure 3.1 Basic authorisation infrastructure model

A variety of research and approaches have been introduced to secure access to organizational resources , such as, role based access control (RBAC) **(INCITS, A., 2004)** and attribute based access control (ABAC) **(Yuan, E., & Tong, J., 2005)** and many other methodologies to help securing this infrastructure and facilitate access control decisions management.

However, most of these approaches are dependent on predefined authorization policies with few mechanisms that are capable to detect illegitimate access and misuse of resources automatically in order to avoid such abuse in the future. **(Bailey, C. et al, 2014).** Authorised administrators, systems auditors and other traditional detection systems may not highly concerned to detect malicious behavior attributed by insiders who might cause much higher level of damage as a result of trust and access rights **(ID Analytics., 2008).** Thus, organizations have to address the authorization and internal security breaches by putting proactive mechanisms in place to reduce and prevent their risk.

3.3 Self-Adaptive Authorisation Framework

One of the most innovative approaches that attempt to address insider threats within authorisation infrastructure are self-adaptive systems through context adaptation. The concept of self-adaptation is oriented with adjusting run time behavior in response with system and environment perception. **(Cheng, B. H., et al, 2009)**. As an example of self-adaptive solution that targets insider threats within authorization infrastructure is SAAF "Self- Adaptive Authorization Framework", which applied self-adaptation concept to manage a federated authorization infrastructure. SAAF controller is implementing reference model that is capable to adopt authorization assets , such as, "policies and subject privileges" whenever a malicious behavior identified within federated authorization infrastructure. However, self-adaptive systems like SAAF approach are facing a challenge to ensure the accurate identification of known and unknown attacks which suggests the application of other anomalous detection approaches as a development of this technology. **(Bailey, C. et al, 2014)**

3.4 Simulating insider attacks through a game

For the purpose of such projects, using historical records of insider attacks could be limited when evaluating self-adaptive authorisation systems. It is essential to have run time data that could reflect both malicious behavior and corresponding self-adaptive response. Hence, in order to simulate insider attacks problem within authorization infrastructure, a game data in a form of log files has been provided by SAAF project. The obtained log files are based on an ethical game of hacking through Snakes and Ladders and they include:

1. Access log, which is a log of access generated by an authorization service.
2. Adaptation log, which is a log of adaptations taken by an autonomic controller, in light of observing access and resource activity.
3. Controller log, which is a detailed log of what the autonomic controller has been doing.
4. SQL logs, which is the data collected from the resource (the game) and stored in SQL format. They contain SAAF database information composed of authorizations, authentications, game, move, roll and users logs.
5. Tab logs, which is the data collected from resources.

The figure below **(Bailey C., 2013)** illustrates the layout of the game supported be the rules, game logic and controls. Players are restricted to certain rules that represent normal behavior of the game.

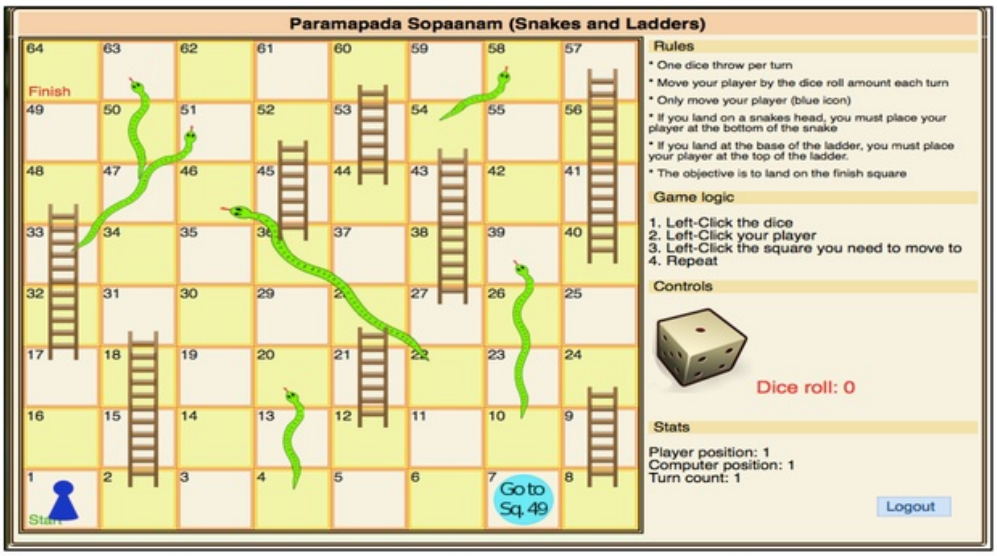


Figure 3.2 Snakes and Ladders game environment

It can be observed from above figure that the ethical way to play this game is to start from position 1 and throw the dice once per turn. Each time, the player should move according to the dice roll value shown on the screen, where the player (blue icon) should land on the right position. If the player has landed on snake's head , such as 20, he must move back to bottom of the snake which is land number 4. Correspondingly, if the player landed on the base of the ladder , such as, land 12, he must move forward to the top of the ladder , such as, land 22. Furthermore, the player might get a bonus to go to the position 49 if landed on land 7. In order to win the game, the player should land on the finish square at position 64. **(Bailey C., 2013)**

The link of this game[1] has been distributed to a number of students to play and try to beat the game via ethical hacking. Each player's information is linked to their authentication account, where authorisations to access game resources are given to that specific username after identity and access rights verification. The authorization, authentication and game logs were recorded accordingly.

[1] https://saaf-resource.kent.ac.uk/game/index.php

It was really essential to start by understanding and analysing the available data by looking to the structure of game data to extract and analyse the relevant data to the research study. SQL logs contain the game complete database that records player's usernames, actions, authorizations, authentications and game information. Each table of the database contains some detailed associated attributes that represent all recorded events in the game. The current database is composed of six main tables, which are:

i. Game Table, it holds the game information , such as, its id, user id, start and end times.
ii. Authorizations Table, it holds authorizations user, identity information, action, decision taken and time requested.
iii. Authentication Table, it relates user to his authentication id and IP address. It is also a field that represent if authentication was successful.
iv. User Table, it holds user information, game won and game lost.
v. Move Table, it holds move id, corresponding roll id, authorisation id, also start and end positions.
vi. Roll table, it holds roll value and roll id with its respected game id and authorisation id.

For the resource level, the game and its corresponding usernames, move, roll information will be quiet sufficient to track some abnormal behavior events recorded by the database logs. The same applies to authorisation table that holds actions and related decisions which gives some possible attacks at this level.

3.5 Detecting Malicious Behavior

The most reasonable way to look into the situations where malicious threats might be detected is to have some assumptions about kind of malicious behaviors could be detected at environment (game level) and other types of misuses could be captured at authorization level. Then, look for a means to simulate those activities at those levels and analyse the possible abuses admitted at both levels.

Thus, some scenarios of the problem could be constructed to hypothesis some anomalies and illegal activities that could exist in this game which might be as follows:

- If the player throws the dice, then he does not land on the right position as per dice roll value.
- If the player throws the dice consecutively before he actually moves, to select his dice value preference.
- If the player moves without rolling the dice.
- If the player injects illegal roll value outside the normal range (0 to 6).

There are also some other illegal activities that could be captured at authorisation level which might be as follows:

- If the players admits any action in the game without getting the right authorization to do so.
- If irregular action or attempt has been logged.
- If the number of authorization request denies exceeds a certain limit.
- If the user acquired authorizations, but misuse them by replicating their actions or use them to do something else.
- If user reuses somebody else authorizations illegally.

Based on the game and its authorization logs, some features similar to the discussed scenarios should be extracted to enable anomalies identification within application (resource) environment and at authorization level. SAAF was successful in detecting some identified threats through observing the execution of a target authorization infrastructure and analysing operational states of the anomalous user behavior. As a result, the authorization decisions were adopted to halt some malicious behavior **(Bailey, C. et al, 2011)**. However, SAAF solution is limited to the predefined and known threats within its target domain model , such as, the management decisions. Management decisions are restricted only to what can be monitored and controlled. Some illegal game actions that happened within game environment may be permitted by SAAF because it is treating game (protected resources) and authorisation (authorisation services) independently with different sets of rules. Consequently, Some malicious behaviors are not currently captured by SAAF because the patterns of such malicious behavior have no rules identified by SAAF to trigger adaptation.

3.6 Summary

Self-adaptive systems , such as, SAAF are effective in reducing the insider's threats malicious behavior identified within federated authorization infrastructures but they are limited to known attacks only. SAAF controller should be merged with other advanced intrusion detection systems such ac machine learning techniques to analyse the abuse at both resource and authorization levels. The following chapter is designed specifically to investigate possible solution for this problem through machine learning approach.

Chapter 4

Solution Domain

4.1 Introduction

As revealed in the previous chapter, self-adaptive systems that target insider threats are quiet limited to known and predefined attacks only. It is also facing a challenge to ensure the accurate identification of both known and unknown attacks. In parallel to this, some studies have been directed to machine learning techniques to investigate their valuable contribution to improve the performance of internal attacks detection. This chapter will discuss machine learning concept and algorithms that could be applied to the problem domain. Moreover, it will evaluate some existing tools and select the most suitable one for experiments. Finally, last section will discuss the application of machine learning to the problem domain in context to SAAF and game environment.

4.2 Machine learning and algorithms

Machine learning is defined broadly as "computational methods using experience to improve performance or to make accurate predictions" **(Mohri, M., et al, 2012)**. In order to explain how machine-learning technology works, Stephen Dodson summarized that by saying, "machine learning works by continually establishing baselines for comparison". Whenever new behaviors observed, machine-learning tool will check if there is any real anomalies detected on regular basis. Even if there is abnormal behavior that is perceived later by organisation to be normal, machine learning classification tools will learn so and behave accordingly in the future. However, machine learning techniques are powerful in running analysis for large amount of data on run time environment with instant detection of malicious behavior **(Dodson, S., 2014)**.

Data mining tools are capable of doing several tasks , such as, classification, regression, clustering, summarization, dependency modelling and deviation Detection **(Kantardzic, M., 2011)**. This project will focus mainly on classification task in order to evaluate the effectiveness of using machine learning to detect insider threats in

access control systems. Classification is basically a predictive learning function, constructed based on the labelled training data, that is capable of classifying data according to the predefined class or classes. **(Witten, I. H., & Frank, E., 2005)**

In order to achieve the project goal and select a proper machine-learning classifier, which can be used to detect anomalies in user behavior, some algorithms have been evaluated. Since the training and testing data which will be provided for the classifier are labelled, this research will just focus on supervised learning algorithms. There are four main algorithms fall under single classifiers category as discussed in literature review that can be briefed below:

i. Naive Bayes, it is a learning algorithm that reduces learning complexity by "making a conditional independence assumption" to reduce the number of parameters to be estimated when modelling the probability. (**Mitchell, T. M., 2005).**

ii. Decision Trees, Decision tree algorithm is a predictive model that maps attributes observations to their prediction conclusions in a structural manner (**Mathew, S., et al, 2010)**. The tree is constructed during the training process as construction is dependent on attribute characteristics observed from the training instances. It can be ordered based on their information gain (**Mitchell, T. M., 2005).** The main example algorithm on decision trees is called J.48 **(Witten, I. H., & Frank, E., 2005).**

iii. Support Vector Machines (SVM), it support vector machines classify instances by constructing "hyperplanes" for each class and then draw a decision boundary during training phase to separate individual classes. The construed decision boundary is used to classify new instances throughout testing phases. The main training instances which form the "hyperplanes" and define decision boundary are called support vectors. (**Mathew, S., et al, 2010).** Using only a selective training samples rather than using the whole dataset, makes the SVM more robust to outliers. Another advantage of using SVM is getting what is called "penalty factor" which enable users to maintain a balance between the number of misclassified instances and the width of defined decision boundary. (**Tsai, C. F., et al, 2009)**

iv. K-Nearest Neighbour, the k-Nearest Neighbour is a classical approach that looks for k instances in training data that are very close to the closest testing

data. As a result, it will assign the most recurrent label among these instances to the new instance. "The only free parameter is the size k of a neighbourhood". **(Laskov, P. et al, 2005)**

Based on above algorithms characteristics, the simplicity and attributes independency of Naïve Bayes algorithm makes it a good approach to consider for this project as it will be investigated in the next section. More sophisticated and enhanced algorithms could be applied to enhance the classification results for future studies.

4.3 Naïve Bayes Algorithm

Naïve Bayes is a simple machine learning approach that relies on using learning probabilistic knowledge to build up its predictive models. Many people who struggled in using sophisticated learning algorithms, realised eventually that Naïve Bayes can performs better or just as well as others. This algorithm is effective when attributes are independent from each other in a given class, which limits its performance where dependent features exist. Other restriction of Naïve Bayes is "normal distribution assumption for numeric attributes". However, using discretization process could solve this or by applying "stranded estimation procedures". **(Witten, Ian H. et al, 2011)**

Naïve Bayes has got several advantages over other classifiers, as it requires much less CPU and memory consumption due to its simple computations. Furthermore, it is robust to irrelevant attributes and to isolated points of noise. It can handle also missing values by discarding the instances during probability estimate computations. In addition, Naïve Bayes is a simple algorithm that assumes the independence between attributes, but still "Bayesian Belief Networks" can be used to handle dependence cases. (**Calders, T., & Verwer, S., 2010**).

The posterior probability is calculated using Bayes theorem from class prior probability P(c|x), predictor prior probability P(x) and the likelihood parameter P(x|c) which represents the probability of predictor given at a certain class. The class with the highest posterior probability is the prediction outcome. The Bayesian equation is shown below **(Sayad, S., 2012)** and the information flow is illustrated in the figure followed (**Vryniotis, V., 2014**):

$$P(c|x) = \frac{Likhood * Prior}{Evidence} = \frac{P(x|c)P(c)}{P(x)} \quad (4.1)$$

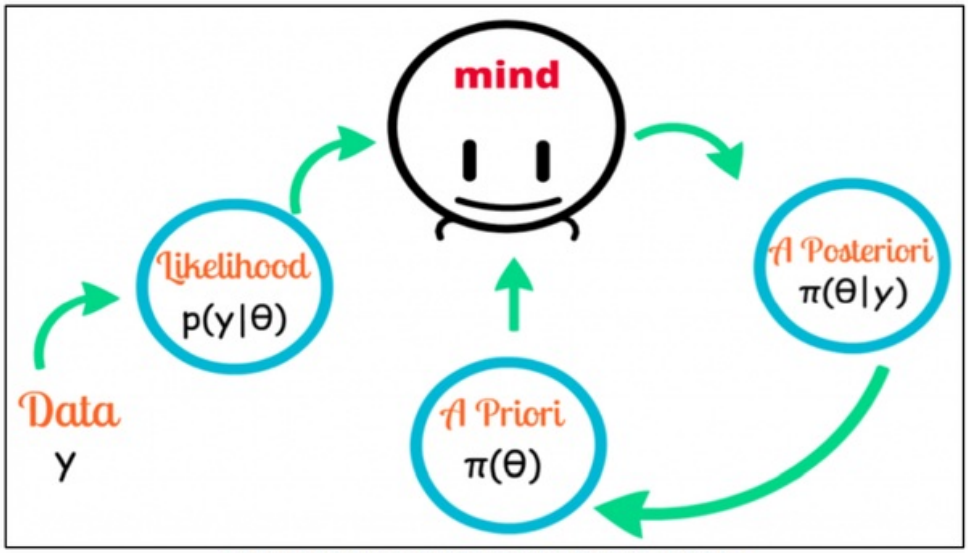


Figure 4.1 Naive Bayes classifier prediction workflow

4.4 Naïve Bayesian Tools

This project is not intended to implement Naïve Bayes algorithm from scratch, as there are many ready tools available to test detection experiments on. In order to identify an effective tool that implements Naïve Bayes algorithm and suit project requirements in terms of abuse classification and analysis, a number of tools have been surveyed as summarized in Appendix II. The selected tool is called Waikato Environment for Knowledge Analysis (WEKA).

WEKA is java-based software **(Cs.waikato.ac.nz, 2005)** that contains a collection of machine learning algorithms that are designed to perform many data mining tasks. It is capable to develop the new machine-learning scheme **(Hall, M., et al, 2009)**. WEKA has been recognized as a "landmark system" over years across machine learning research communities and in data mining domain "because it is the only toolkit that has gained such widespread adoption and survived for an extended period of time" **(Markov, Z., & Russell, I., 2006)**. The algorithms can either be applied directly to a dataset or called from your own Java code. WEKA can provide both Graphical User Interface (GUI) and command line interface with a good range of built in functionalities , such as,: "data pre-processing, classification, regression, clustering, association rules, and visualization" Furthermore, it enables users to test

and compare different machine learning algorithms on their data sets. **(Hall, M., et al, 2009)**.

WEKA is well-used tool for Naive Bayes algorithm as it can handle continuous attributes and model them "as a single normal" **(Witten & Frank, 2005).** After testing WEKA with a synthetic data of project inputs, it has been decided that WEKA could suits the purpose of this project where it is capable of pre-processing, classifying, analysing and evaluating the features of user's behavior. In addition to the classification output, WEKA provides a summary of the tool predictions versus the actual supplied expectations. Moreover, it can facilitate features selections task where features have to be filtered according to the classification relevance by making use of "Evaluate Attributes" option which is ranking attributes based on many evaluator's alternatives and search methods. Hence, this tool will be used throughout all experiments stages that involve machine learning detection.

4.5 Application to the problem domain

Throughout the discussion of the problem domain in the previous chapter, there are some possible attacks that could exist in both game level and authorization level of SAAF. This section will give an overview of the application plan of machine learning to the game.

4.5.1 Classification prerequisites

In order to apply machine-learning approach to the game, some malicious behavior activities should be captured in a form of features or attributes that are represented by meaningful numeric value. These values should be well understood by Naïve Bayes which will ultimately calculates the probability of an event instance to be flagged as abuse or not based on the training data injected to WEKA tool.

4.5.2 Data processing

Based on data provided by the game activity logs, the fundamental phase is to extract many features that represent any malicious behavior formulated by breaking the game rules in both application and premises levels. These features have been extracted by running different SQL queries, as feature extractor, starting from application features that are based on player's behavior during the game play and then combining them with additional features extracted from the access logs that are

generated using authorization service of SAAF. Each group of features was plugged-in and analysed separately using Weka machine learning tool that classify the features (as attributes) as per the selected Naïve Bayes algorithm. It is important to clarify that complete dataset have to be spilt into training and testing sets, where most of the data should be used for training to discover predictive relationships between attributes. Whereas, testing dataset is used to assess the effectiveness and strength predictive relationships. Figure 4.2 **(Nltk.org, 2014)** illustrates that training dataset contribute to build the classifier model through the specified machine learning algorithm, whereas prediction (test dataset) will be classified with the means of this classifier and labelled as abuse or not accordingly.

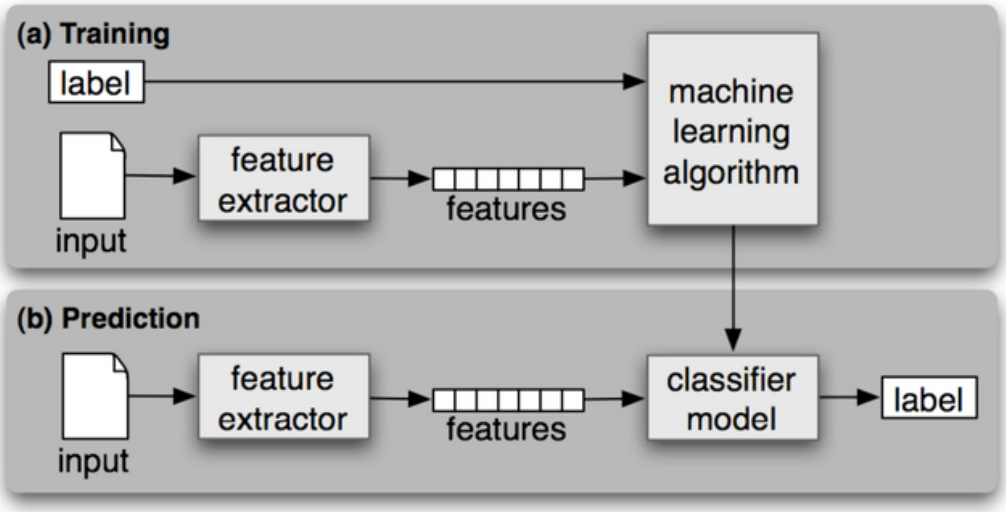


Figure 4.2 Flow of training and prediction datasets through machine learning

4.5.2 Classifiers evaluation

By running Naïve Bayes classifier, the accuracy of the classification and performance measures can been observed to judge the significance of the selected features and accuracy rates obtained by the classifier. The selected features and attributes of attacks were evaluated to test the relevance and the worth of each of them in the classification phase. Eventually, the selected attributes from the application level were combined with authorization level features and injected to the tool in order to detect possible abuses originated from the mixture of features of two phases , such

as, effectiveness and limitation of SAAF approach. The ultimate objective of combining the two sets of features (environment and authorization level features) are utilized to check if they complement each other through running a single classifier on them. Then, according to analysis results classifiers result could be fused together to make the final judgement on malicious behavior activities whether players have abused their authorisations. Each phase of experiments resulted in a training model that already learned from its supplied data and abuse patterns figured from extracted features. Hence, those models can be used any time to validate the results by plugging any game live data.

4.6 Summary

Machine learning techniques got the power of learning mechanism that looks for anomalies behavior constantly and updates its systems with its new discoveries about malicious behaviors observed. Naïve Bayes is a good algorithm in the context of game due to its simple computations, robustness to irrelevant attributes and features independence throughout learning process. Moreover, it can deal with the uncertainty presented with extracted features throughout learning stage.

WEKA tool is a flexible machine learning tool that could run Naïve Bayes and other learning algorithms with a display accuracy and performance measures that facilitate features selection, malicious behavior analysis and phases evaluations that makes it highly recommended for this project. However, the applicability of the solution has been experimented in the next chapter throughout multi phases of experiments as suggested in sections 4.5, on game data, authorisation data and combined set of data.

Chapter 5

Experiments

5.1 Introduction

While carrying out this project, the work has been sub divided into four main phases to facilitate data handling and assure more accurate classification and detection of the most relevant forms of attacks in game and its authorisation level. In conducting each phase test, the features (attributes) were classified using Naïve Bayes classifier as per as an additional a class attribute called *abuse* to test if the selected features or the combination of features detection indicates an abuse of the resources or not. The analysis of each phase involves the discussion of classifier performance measures and relevance of selected attributes to the classification results. Hence, this chapter will start overviewing the main data mining performance measures that evaluates the accuracy of actual datasets against algorithm predictions followed by detailed analysis of each phase results beside the discussion of their corresponding measures. Phase 1 is concerned about classifying the features of game environment and phase 2 is handling the features that represent malicious behavior at system (authorisation) level. Then, phase 3 of experiments is combining both game environmental features and authorisation features to scope the internal malicious behavior at both levels simultaneously. Phase 4 is just fusing the independent classifier results of phases 1 and 2 using simple OR gate to extend the detection scope of malicious behaviour.

5.2 Machine Learning Performance Measures

As long as this project is dealing with different forms of insider attacks, it is essential to evaluate them using machine learning with various quality and performance measures to assess their effectiveness and weigh their influence ratio to the detector classification and decision taken whether user behavior is abuse or not.

Weka is like other machine learning tool that provides the user with most common accuracy details and predictions summary (number of correct predictions out of all predictions made) which helps in evaluating the robustness of a classification model. However, in order to build a model for a classification problem, the accuracy

of the model is not enough to evaluate the effect of provided attributes in this current classification and make a decision whether the predictive model is capable enough to solve the problem. Moreover, below is an overview of the most relevant machine learning performance measures that can be extracted from WEKA tool output which are as follows:

- TP Rate, it is the rate of true positives which represents the number of instances that are correctly classified in a given class out of the full number of instances.
- FP Rate, it is the rate of false positives which represents the number of instances that are falsely classified in a given class out of the full number of instances.
- Precision, it is the proportion of instances that are truly classified and divided by the total number of instances classified as that class. **(Bouckaert, R. R., et al, 2013)**
- Recall, it is proportion of instances specified for a given class divided by the actual total in that class (number of True Positives divided by the total number of True Positives and False Negatives). Moreover, some analytics use recall to measure the sensitivity and the classifier completeness, as it is focusing on the percentage of positive instances **(Jasonb, 2014)**
- F-Measure, it is A combined measure for precision and recall which can be calculated by

A weighted mean of precision and recall = $\frac{2\ x\ Precision\ x\ recal}{(Precision + Recall)}$ (5.1)

- Receiver Operating Characteristic (ROC), it evaluates the performance of the classifier at a varying threshold. The ROC value represents the area under the curve that plots a fraction of true positives out of the total actual positives versus a fraction of false positives out of the total actual negatives **(Fawcett, T., 2004)**. Basically, an "optimal" classifier will have ROC area values approaching 1, with 0.5 being comparable to "random guessing". **(Swets, J. A., 2014)**.

Moreover, Weka is provides error summary table that combines most of the above measures in a logical paradox called "Confusion Matrix". It comes in the following format which describes how predictions are tabulated:

Table 5.1 Confusion Matrix layout

		Predicted Label	
		Positive	***Negative***
Known Label	***Positive***	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

The confusion matrix layout is basically a detailed summary of the classifier accuracy where all components contributes in calculating the overall accuracy , such as, the percentage of correct predictions equals to

$$\frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (5.2)$$

A more comprehensive contingency matrix detailing all the measures analysis that can be obtained from the confusion matrix is provided in *Appendix I*. **(Stehman, S. V.,1997).**

5.3 Classification using game application level data

5.3.1 Objectives

The ultimate goal of this step is to work on the game data, its logs and extract some application (resource) features that represent a form of user attempts to attack the game by violating the game rules and policies or through injecting glitches to the game. By the end of this phase, some meaningful and relevant features (on resource level) should be extracted and classified using Naïve Bayes algorithm to assess their influence on the detector decision whether user behavior is classified as abuse or not.

5.3.2 Method

Based on input logs of the game activities, some features have been extracted from the application layer. Initially, two malicious behaviors were surveyed by focusing on the player movement in the game, which are as follows:

i. Number of consecutive dice rolls before an actual move (RD)

ii. Number of places where piece moved beyond the dice roll value (MP), which reflects a behavior where player has not landed on the right position.

The features were extracted using SQL (Structured Query Language) by applying special queries, as documented in *Appendix III (Experiments 2 and 3)*, that helped in discovering the points of violations where users attempt to break the game rules and behave maliciously. Then, all features were combined in a common table along with their associated information and details of each user and game in order to facilitate behavioral analysis. Afterwards, decisions have been made on the selected features with the combinations of such features forms an abuse of the system or not. A specific, an agreed conditions and thresholds were set for each feature to decide whether such behavior is classified as abuse or it is just a normal behavior. Primarily, the classifier was trained to classify each combination of features as abuse if one of features is misuse which means that one violation of game rules could leads to an abuse decision from the detector. Finally, based on provided set of quality and accuracy measures of WEKA machine learning tool, the attributes (features) where evaluated to test the effect of features and the combination of features (both RD and MP) on the classifier predictions.

5.3.3 Results and Discussion of Results

First of all, two features were extracted as discussed above and judged in the sense of:

- If illegal move value (MP) is not equal to 0, it is indicates that the user does not land on the right position.
- If the number of the consecutive rolls (RD) is not equal to 1, this will be trained as abuse.

Prior training the classifier with the features, the abuse decision has been made automatically "YES" or "NO" if one of two features becomes true for "YES" and "NO" for normal values of the features, where NO reflects the absence of abnormal behavior. Those decisions were analysed on users based , such as, the values were related to each user who attempted the attack. Hence, three main attributes is pre-processed into machine learning tool, which are:

i. Illegal Move Value of the Piece (MP)

ii. Number of Dice Rolls Before Move (RD)

iii. Abuse class

By injecting the extracted two features values dataset into WEKA with 70% split percentage (where the rest of 30% will be used for testing), the following evaluation summery was obtained as per Naïve Bayes Classifier:

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                    Class
Attribute             NO    YES
                   (0.79) (0.21)
==================================
ILLEGALMOVE_Value
  mean                 0 -0.141
  std. dev.       1.2222 8.7273
  weight sum         601    156
  precision       7.3333 7.3333

rollsBeforeMove
  mean            1.1667 2.3408
  std. dev.       0.1944 2.2085
  weight sum         601    156
  precision       1.1667 1.1667
```

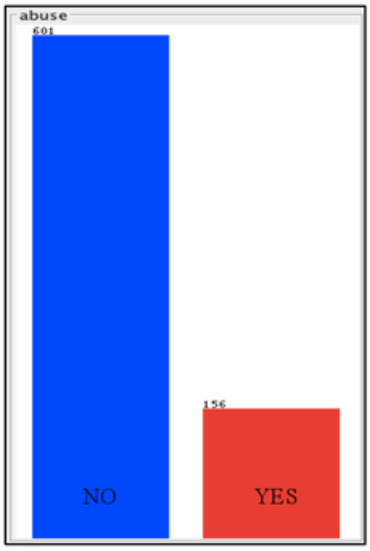


Figure 5.1 Classifier model output

It can be seen from above figure that around 21 % of the dataset has been caught by the classifier as malicious behavior, whereas 79 % of the dataset has been classified as normal behavior. Moreover, the above output clearly reflects that the first feature represents distance moved beyond the roll dice value with much higher precession (7.3333) than the second feature that count the number of consecutive rolls thrown before an actual move which comes with only 1.7692 precision and this is reflects the value of the first feature on the classifier decisions sue to its better indications.

Furthermore, out of 757 instances provided from game log, 227 instances (30 % of the dataset) have been used for testing were the rest of the data have been used to train the classifier which forms the remaining 70 % of the dataset. After the classifier running, the following classification and performance results were obtained as shown in figure below:

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         192               84.5815 %
Incorrectly Classified Instances        35               15.4185 %
Kappa statistic                          0.4408
Mean absolute error                      0.1716
Root mean squared error                  0.3841
Relative absolute error                 50.8224 %
Root relative squared error             90.4363 %
Total Number of Instances              227

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.66       0.833     1         0.909      0.67     NO
                 0.34      0          1         0.34      0.507      0.67     YES
Weighted Avg.    0.846     0.506      0.872     0.846     0.815      0.67

=== Confusion Matrix ===

   a   b   <-- classified as
 174   0 |   a = NO
  35  18 |   b = YES
```

Figure 5.2 Classifier evaluation of test dataset

With the reference of above figure, it can be observed that out of 227 test instances, 192 were correctly classified with the percentage of 84.5815 accuracy rate whereas, the remaining 35 were actually incorrectly classified based on naïve bays classifier. After comparing the actual classification versus tool classification, it was observed that whenever the data is close to the pre-defined thresholds, the classifier is sometimes record some of them as misclassifications. This reflects that there is a level of uncertainty in Naïve Bayes classification because sometimes classifier is not certain exactly on the specified desired ranges and could leads to some false positive and true negative predictions.

However, the detailed "by class" accuracy summarizes the performance measures of the classifier and in this experiment can be briefed as follows:

- TP Rate (TPR), the rate of true positives is equal to 1 which reflect that all instances correctly classified as for NO class and it is 0.34 for YES class because only 32 instances were truly classified as YES.
- FP Rate (TFR), the rate of false positives is equal to 0.66 because 35 of instances were falsely classified as YES and it is 0 as there is no instances falsely classified as YES.
- Precision, the NO class has 0.833 precision, whereas YES class achieved 1 precision because of the accurate classification of all instances as YES.
- Recall, the NO class has 1 recall which directly calculated by dividing the number of True Positives (174) with the number of True Positives (174) and

the number of False Negatives (0). Whereas, YES class achieved 0.34 recall which is calculated in similar way to NO class.

- F-Measure, 0.909 for the combined measure of precision and recall of No class and 0.507 for YES class.
- ROC, the area under the curve is equal to 0.67 for both YES and NO classes.

Moreover, the confusion matrix summarizing the error matrix as shown in following table:

Table 5.2 Confusion matrix of first application features

174 true positives (non-abuse that were correctly marked as NO)	**0 false negatives** (abuse that were incorrectly marked as non-abuse)
35 false positives (non- abuse that were incorrectly marked as abuse)	**18 true negatives** (correctly classified abuse YES)

Based on above discussion of the performance measures and values obtained for each individual, it seems that the most relevant and useful measures for the project classification would be the precision and recall as a result of their good interoperation of predictions made by the classifier. For example, higher precision indicates lower false positive rate, while a lower precision indicates higher false positive rate. In other words, precision reveals the exactness of the classifier. On the other hand, recall measures the completeness of the classifier. Higher recall means that there are less false negatives, though lower recall means having more false negatives.

One way to improve the performance of game features classifier is to increase the number of attributes contributing in the final classification of the behavior. Thus, phase 1 has been extended to cover more features, as documented in *Appendix III (Experiment 9)*.

- **Phase 1 extension:**

In order to enhance the accuracy of the classifier and have a larger scope of detection, two more features were extracted and combined with MP and RD features, which are:

i. The accumulative value of illegal move distance value of the piece.

ii. The accumulative value of number of dice rolls before move.

Basically, they are counting the occurrence of earlier discussed two features. Such type of attacks are correlated to each other and the accumulative values of them could be compared to some thresholds to judge if user really abuse the reflected gained resources by exceeding one or both limits of attempts. For this particular experiment, the detector has been trained to classify the instance if one of the following scenarios (form of misuse) occurs:

i. Distance moved beyond or less dice roll (MP) is not equal to zero.
ii. The number of rolls before move (RD) greater than one.
iii. The accumulative value of MP (CUM MP) is greater than 21.
iv. The accumulative value of RD (CUM RD) is greater than or equals to 10.

The above extracted four features were exported to a comma separated file so that they can be injected to the tool which automatically read them as *.arff* format which is the main file format of WEKA inputs. The classifier was operated again with the four provided features (attributes) and the evaluation summary of the performance measures was observed as shown in below screenshot of WEKA output.

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         195               85.9031 %
Incorrectly Classified Instances        32               14.0969 %
Kappa statistic                          0.6007
Mean absolute error                      0.1466
Root mean squared error                  0.3658
Relative absolute error                 38.6083 %
Root relative squared error             81.7958 %
Total Number of Instances              227

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.97      0.435      0.856     0.97      0.909      0.834    NO
                 0.565     0.03       0.875     0.565     0.686      0.834    YES
Weighted Avg.    0.859     0.325      0.861     0.859     0.848      0.834

=== Confusion Matrix ===

   a   b   <-- classified as
 160   5 |   a = NO
  27  35 |   b = YES
```

Figure 5.3 Classifier evaluation test results of four application features

It can be seen from the above figure that the accuracy of the classifier has been enhanced a bit to 85.9 % with 14 % as misclassification. It was observed that False Positive Rate is higher for the NO class. By taking a subset of the predictions to investigate the uncertainty thresholds of the classifier, it was observed that Naïve

Bayes has a higher false positive rate for the "NO" abuse class where it reports sometimes an error, represented by + sign with 0.988 probability that those instances are incorrectly classified as abuse whenever first feature (MP) is just below the threshold as shown in the figure below.

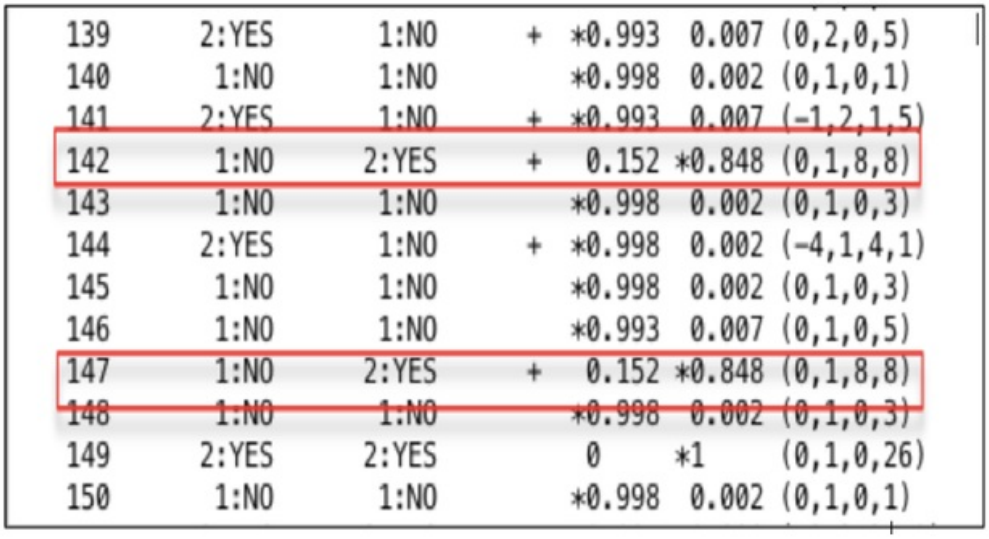
```
=== Predictions on test split ===

inst#,    actual, predicted, error, probability distribution
     1       1:NO       1:NO          *0.973  0.027 (0,1,1,7)
     2       1:NO       1:NO          *0.998  0.002 (0,1,0,1)
     3       1:NO       1:NO          *0.998  0.002 (0,1,0,1)
     4       1:NO       1:NO          *0.998  0.002 (0,1,0,1)
     5       1:NO       1:NO          *0.998  0.002 (0,1,0,1)
     6       1:NO       1:NO          *0.998  0.002 (0,1,0,1)
     7       1:NO       1:NO          *0.998  0.002 (0,1,0,3)
     8      2:YES       1:NO      +   *0.998  0.002 (-2,1,2,1)
     9      2:YES       1:NO      +   *0.998  0.002 (-1,1,3,1)
    10       1:NO       1:NO          *0.998  0.002 (0,1,0,3)
```

Figure 5.4 Actual prediction vs. classifier predictions on test split

In order to investigate the mechanism the machine-learning tool learns new malicious activities throughout its learning process, it was noticed that the classifier has detected some new anomalies and marked them as abuse. For example, by looking deep into the below subset if:

- The distance moved beyond or less dice roll (MP) is zero,
- The number of rolls before move (RD) is one,
- The accumulative value of MP is 8,
- The accumulative value of RD is 8.

The classifier learns that this behavior forms an abuse for the resource over time.



139	2:YES	1:NO	+	*0.993	0.007	(0,2,0,5)
140	1:NO	1:NO		*0.998	0.002	(0,1,0,1)
141	2:YES	1:NO	+	*0.993	0.007	(-1,2,1,5)
142	1:NO	2:YES	+	0.152	*0.848	(0,1,8,8)
143	1:NO	1:NO		*0.998	0.002	(0,1,0,3)
144	2:YES	1:NO	+	*0.998	0.002	(-4,1,4,1)
145	1:NO	1:NO		*0.998	0.002	(0,1,0,3)
146	1:NO	1:NO		*0.993	0.007	(0,1,0,5)
147	1:NO	2:YES	+	0.152	*0.848	(0,1,8,8)
148	1:NO	1:NO		*0.998	0.002	(0,1,0,3)
149	2:YES	2:YES		0	*1	(0,1,0,26)
150	1:NO	1:NO		*0.998	0.002	(0,1,0,1)

Figure 5.5 A subset of test result with new abuse detected

Eventually, the features of the game were evaluated by utilizing attributes evaluator option from WEKA tool. During classification, *Info Gain Attribute Evaluator* and *Attribute Ranking* method have been selected to evaluate the worth of each attribute by measuring the information gained with respect to the class.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 5 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.252  2 rollsBeforeMove
 0.248  1 ILLEGALMOVE_Value
 0.215  4 CUM_RD
 0.141  3 CUM_MP

Selected attributes: 2,1,4,3 : 4
```

Figure 5.6 Phase 1 attributes evaluation summary

Attributes evaluation output as shown in the figure above, the most important attribute that has a higher ranking is the consecutive number of rolls before move (RD) followed by the illegal move distance value (MP). The least powerful attribute

that came on last rank is the accumulative value of (MP). In order to proof the concept of detecting the malicious behavior at the resource level, all suggested four features will be used from this phase in order to have a wider range of detection scope.

The discussion of this phase was quiet detailed as it forms the base of all upcoming experiments and facilitates the understanding of WEKA tool output and its functionalities. For the next set of experiments, the discussion will be more focused, as just target level will change but the overall methodology will be the same.

5.4 Phase 2: Classification using authorisation level data

5.4.1 Objectives

The ultimate goal of this step is to work on authorization service logs and capture samples of malicious behavior at this level. Then, train and test the extracted features of attacks using Naïve Bayes algorithm and evaluate its effectiveness in capturing those attacks.

5.4.2 Method

By looking thoroughly to the authorisation logs that contains 2340 records, some features that represent misuse of the system resources were extracted and they are as follows:

i. Deny decision count, that represents if the user exceeds a threshold limit of denies, it will be classified as misuse
ii. Count number of consecutive actions, if any action, , such as, start, move, ladder and roll is repeated consecutively and exceeds a certain limit, it will be classified as misuse
iii. Count number of unknown actions, if the number of unknown actions exceeds a certain limit, it will be classified as misuse

Prior injecting the extracted features into Weka machine learning tool, actual predictions were tabulated where the classifier should detects abuse in the following conditions:

- If *Deny Count* is greater than or equal to 5, OR
- If *Count Number of Consecutive Actions* is greater than or equal to 10, OR
- If *the Count of Unknown Actions* is greater than or equal to 3

Afterwards, the dataset of extracted three features, forms of attacks, has been classified using Naïve Bayes classifier. Then, based on provided set of quality and accuracy measures of WEKA machine learning tool the attributes where evaluated using *InfoGain* and *Ranker* method to test the effect and the combination of features (all three features) on the classifier predictions.

5.4.3 Results and Discussion of Results

With the reference to the classification results of authorisation level features shown in figure 5.7, 699 out of 729 test instances were correctly classified with 95.8848 % accuracy rate and naïve bays classifier reflects that the remaining 30 were actually incorrectly classified.

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         699               95.8848 %
Incorrectly Classified Instances        30                4.1152 %
Kappa statistic                          0.8523
Mean absolute error                      0.0547
Root mean squared error                  0.2017
Relative absolute error                 19.4578 %
Root relative squared error             53.8479 %
Total Number of Instances              729

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.977     0.13       0.974     0.977     0.975      0.988    NO
                 0.87      0.023      0.884     0.87      0.877      0.988    YES
Weighted Avg.    0.959     0.112      0.959     0.959     0.959      0.988

=== Confusion Matrix ===

   a   b   <-- classified as
 592  14 |   a = NO
  16 107 |   b = YES
```

Figure 5.7 Classifier evaluation test results of authorization features

The classifier evaluation summary reveals that out of 729 test data there are:

- 592 true positives (actual non-abuse that were correctly classified as NO)
- 16 false positives (abuse that were incorrectly marked as Non-abuse)
- 14 false negatives (Non-abuse that were incorrectly marked as abuse)
- 107 true negatives (all the remaining instances, correctly classified abuse YES)
- High recall, which indicates that there are less false negatives.
- High precision, which indicates that there are less false positives.

Moreover, attributes evaluator has produced the following summary as shown in figure 5.8. It is clearly indicates that *Count Number of Consecutive Actions* is the leading feature in the classification results, whereas *Count number of unknown actions* contributes with much less information during the classification process.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 4 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.4748  2 Count Number of Consecutive  Actions
 0.182   1 Deny Count
 0.0896  3 Count Uknown Actions

Selected attributes: 2,1,3 : 3
```

Figure 5.8 Phase 2 attributes evaluation 1

It can be grasped that authorisation features scores a higher accuracy with the amount of (95.8%) compared to classifier accuracy of game environmental features (85.9)% due to the size of the authorisation training data which is almost double game data (move level features). In addition, authorisation features come with lower false positive rate and higher precision of the prediction class. The above discussed phases were treated independently by the classifier at two different levels, which were a game application level (environment) and authorisation level (system). However, phase 3 will look into a high level integration of these two phases.

5.5 Phase 3: Classification using the combination of application and authorisation data

5.5.1 Objectives

The ultimate goal of this step is to combine both authorization features and game application features into one dataset file and classify them using Naïve Bayes

algorithm as one dataset. Then, evaluate the effectiveness of combined features set to capture malicious attacks.

5.5.2 Method

The working principle of combining two phases features is relying on grouping the seven features extracted throughout above two experimental phases into one dataset and apply the same rules as summarized in the table below:

Table 5.3 Combination of game and authorization features

	Feature	*Level*	*Condition*	*Threshold of misuse*
1	Deny Count	Authorisation	if the user exceeds a threshold limit of Denies	>=5
2	Count Number of Consecutive Actions	Authorisation	if any action is repeated consecutively and exceeds a certain limit	>= 10
3	Count number of Unknown Actions	Authorisation	if the number of unknown actions exceeds a certain limit	>=3
4	RD	Resource	Consecutive number of rolling the dice before actual moves	!= 1
5	MP	Resource	Number of places of moving the piece different that the dice roll value	!= 0
6	CUM MP	Resource	The accumulative value of the number illegal places the user moved to, or failed to move to.	> 21
7	CUM RD	Resource	The accumulative value of the number of consecutive number of rolls before the player moves.	> = 10

Then, the grouped features were classified using one single classifier to visualize potential abuses reflected from both game and authorisation levels in the case of occurrence of a misuse in any of the these levels.

5.5.3 Results and Discussion of Results

While trying to combine both application features and authorisation features, a problem was encountered due some reused authorisation identifications (authz id) in the game log (move table). Some players were having the same authz id associated with different move and roll actions in the game as can be seen in figure 5.9 below:

id	user_id	action	game_id	move_id	roll_id	authz_id	roll_value	decision
1223	mike	roll	103	573	0	17	0	permit
1224	mike	roll	103	575	0	17	0	permit
1225	mike	move	1	8	8	18	5	permit
1226	mike	move	121	585	0	18	0	permit
1227	mike	move	110	581	0	18	0	permit
1228	mike	roll	125	594	0	19	0	permit
1229	mike	roll	125	595	0	19	0	permit

Figure 5.9 Reused authorization ID problem

This problem was figured out because the actual number of records in authorisation table are 2430, however the combine dataset of both game and authorization data consists of total 2443 instances with 13 additional instance (replicated instances). For the illegitimate instances, a player is actually authorised to do something, but he re-used the same ID to perform some malicious action. For example, authorization ID 17 is permitted actually to roll; however the user reused the same ID to move within the game illegally. The possible explanation of this is that the player has carried out a session poisoning attack, where he may injected certain session values which are used when logging to exploit a vulnerability in the game. For this phase experiment, the adopted solution to avoid this issue was forming a distinct authorization id and removing illegitimate authorization ids records. The balance records with (2428) of the combined features were processed to WEKA and classified in the same manner of phase 1 and 2.

With reference to classification results of authorisation level features shown in figure 5.10, 683 out of 728 test instances were correctly classified with 93.8187 %

accuracy rate and naïve bays classifier reflects that the remaining 45 were actually incorrectly classified.

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         683               93.8187 %
Incorrectly Classified Instances        45                6.1813 %
Kappa statistic                          0.8156
Mean absolute error                      0.0656
Root mean squared error                  0.2333
Relative absolute error                 18.4686 %
Root relative squared error             55.2678 %
Total Number of Instances              728

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.986     0.219      0.937     0.986     0.961      0.973    NO
                 0.781     0.014      0.943     0.781     0.854      0.973    YES
Weighted Avg.    0.938     0.171      0.938     0.938     0.936      0.973

=== Confusion Matrix ===

   a   b   <-- classified as
 551   8 |   a = NO
  37 132 |   b = YES
```

Figure 5.10 Classifier evaluation test results of combined features

The classifier evaluation summary reveals that, out of 728 test data, there are:

- 551 true positives (actual non-abuse that were correctly classified as NO)
- 37 false positives (abuse that were incorrectly marked as Non-abuse)
- 8 false negatives (Non-abuse that were incorrectly marked as abuse)
- 132 true negatives (all the remaining instances, correctly classified abuse YES)
- High recall, especially for NO class, indicates that there are less false negatives.
- High precision, which indicates that there are less false positives.

By revising thoroughly the accuracy figures obtained through all the three phases, the following summary is initiated as shown in table:

Table 5. 4 Summary of classifiers accuracy

Phase	*Classifier accuracy measure*
Classification of game features only	85.90 %
Classification of authorisation features only	95.88 %
Classification of combined set of features	93.81 %

It is obvious that the combined set of features is slightly lower than authorisation level features set though they consist of nearly the same number of training and test instances. This is because of the higher false positive rate with amount of (0.219) for the NO abuse class compared to only (0.13) for authorisation only dataset. This is due to the combination effect with game application features that have a higher level of uncertainty near the defined thresholds of malicious behavior recognition, as observed after looking to individual subsets of predicted data.

Furthermore, the combination phase results in a higher number of abuse cases detection. Figure 5.11 visualised the difference in abuse cases, marked by YES red columns, between authorisation features only and the combined set with application level features.

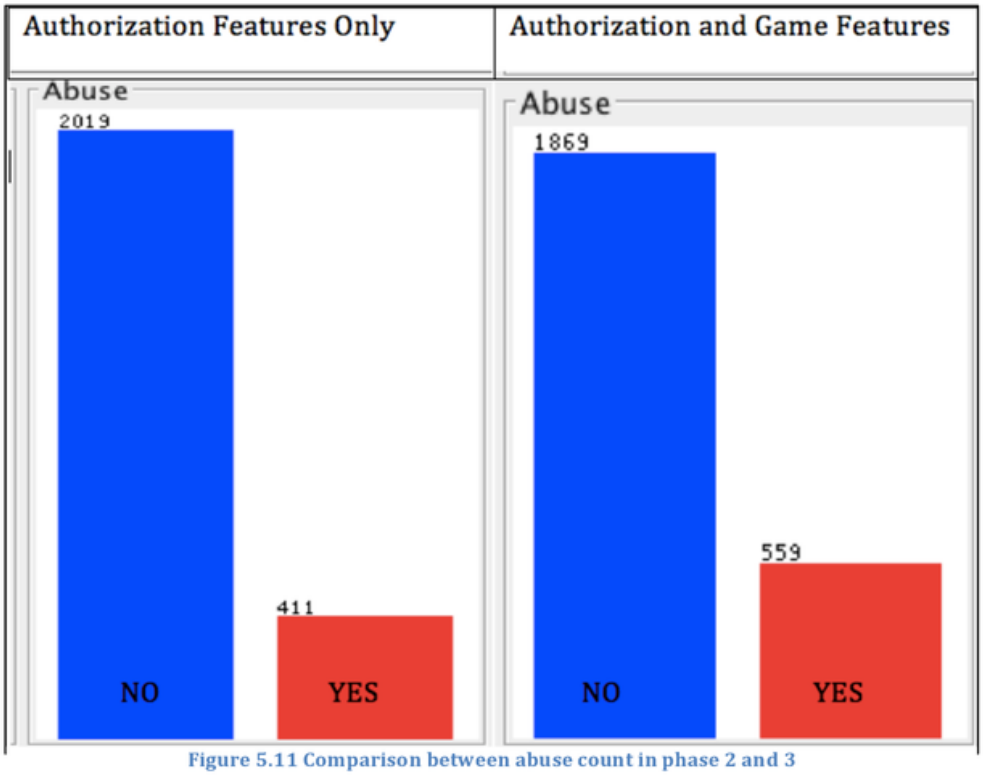


Figure 5.11 Comparison between abuse count in phase 2 and 3

It can be seen from the above figure that the combined table of features offered the opportunity to detect a higher number to the additional of 148 abuses which were not detected in the system level only especially (phase 2). This suggests that looking to both application and authorisation levels gives a wider scope of detection that reveals both sets of features complement each other. The evaluation summary of the "Information Gain Ranking Filter" supports this judgment as both authorisation and game features contributes highly in the classification results as demonstrated in figure 5.12.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 8 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.3707    6 Count Number of Consecutive  Actions
 0.1993    4 CUM_RD
 0.1461    5 Deny Count
 0.1374    3 CUM_MP
 0.0768    2 rollsBeforeMove
 0.0735    1 ILLEGALMOVE_Value
 0.0707    7 Count Uknown Actions

Selected attributes: 6,4,5,3,2,1,7 : 7
```

Figure 5.12 Phase 3 attributes evaluation summary

The above figure illustrates that (Count number of consecutive actions), authorisation level feature scored the top rank followed by (accumulative value of number of dice rolls before move). The application level feature where these all together influence the classifier prediction's decision. After analysing the experimental results obtained from all three phases, it was observed that the machine-learning tool is facing a challenge of false positive predictions which could be considered as limitation of this mechanism and has to be addressed to enhance the confidence about classifier predications results.

5.6 Phase 4: Fusing classifiers results

5.6.1 Objectives

The aim of this phase is to fuse the independent classifier results obtained from phases 1 and 2 using simple OR gate as one way to fuse the classification results to assure whether the behavior is malicious or not and observe the performance of first two classifiers collaboratively.

5.6.2 Method

The common instances between the two classifiers were extracted and their classification results were fused together using simple logic OR gate as overviewed in the following figure:

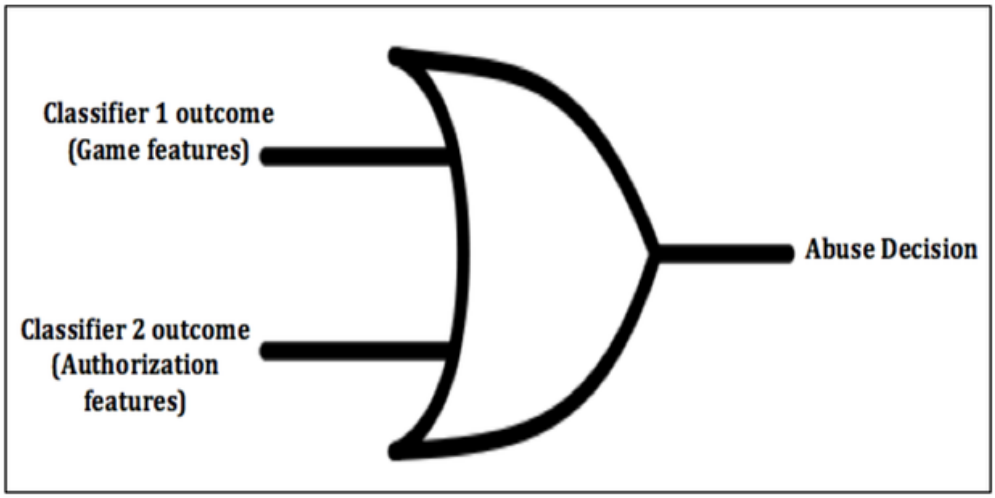


Figure 5.13 Fusing game and authorization features classifications using OR gate

5.6.3 Results and Discussion of Results

The common instances fused together from the first two phases were found to be 757 instances including 227 as test instances. The following table contains a small subset of the fused classifiers results

Table 5.5 A subset of fused classifiers result

INDEX	*user_id*	*authz_id*	*action*	*decision*	*Classifier 1*	*Classifier 2*	*Abuse*
1	rk308	1248	move	permit	NO	YES	YES
2	rk308	1250	move	permit	NO	YES	YES
3	rk308	1254	move	permit	NO	YES	YES
4	rk308	1257	move	permit	NO	YES	YES
5	rk308	1259	move	permit	NO	YES	YES
6	rk308	1261	move	permit	NO	YES	YES
7	rk308	1263	move	permit	NO	YES	YES
8	rk308	1265	move	permit	NO	YES	YES
9	rk308	1269	move	permit	NO	YES	YES
10	rk308	1272	move	permit	NO	YES	YES
11	rk308	1276	move	permit	NO	YES	YES
12	rk308	1278	move	permit	NO	YES	YES
13	rk308	1280	move	permit	NO	YES	YES
14	rk308	1282	move	permit	NO	YES	YES
15	rk308	1285	move	permit	NO	YES	YES

Then by fusing the outcomes of game features and authorization features' classifications, and by applying simple OR gate on the results. It has been found that out of 227 test instances:

- Game features came with 39 abuses.
- Authorization features came with 93 abuses.
- Fused OR gate outcome came with 118 abuses originated from both cases.

The total number of abuses in the fused table reflects that there is 14 common abuses captured by both classifiers. By considering the seven selected features, the common abuses might originated where the player rolls the dice consecutively as this is also captured by "Count number of consecutive actions" authorization feature.

By going through the full subset, it was observed that two set of features (application and authorisation) are complement to each other in a way that if the malicious behavior has not been captured by the first classifier results, it has been captured by the second classifier leading to more accurate decision whether this behavior is abuse or not. For example, by looking to the above table, the violation

adopted by performing the same action consecutively (move) has been not detected by game context features. However, the second classifier of authorisation level features has grasped this violation resulting in "Abuse" classification of the overall behavior.

A full set of experiments conduct throughout this project can be found in Appendix III.

5.7 Summary

The experimental analysis has been divided into main four phases:

- Phase 1: *Classification using game application level data*
- Phase 2: *Classification using authorisation level data*
- Phase 3: *Classification using the combination of application and authorisation data*
- Phase 4: *Fusing game and authorisation classifiers results*

Phase 1 contributes in detecting the malicious behavior in context of the game (resource level), whereas phase 2 contributes in detecting the malicious behavior at the authorisation level through a set of some features of misuse extracted from application and authorisation logs. Phase 3 offered a wider range of detection by helping the classifier to detect higher number of attacks than each phase independently. Phase 4 improved the accuracy of the abuse classifier by fusing the first independent two classifiers through a simple OR gate, as both set of features complement each other. The first three classifiers have been evaluated based on machine learning performance measures. Precision and recall gave a better indication of results along with confusion (error) matrix. All features have been evaluated to test the significance and relevance to the classification process, where (consecutive number of repetitive actions) seems to be the major contributor of malicious behavior. By inspecting some data subsets and their classification, it was observed that learning process adjust the classifier decisions on whether the malicious behavior formed by a set of attributes is really abuse of resources or not. On the other hand, after analysing the experimental results, it was observed that machine-learning tool is facing a challenge of false positive predictions which could be considered as limitation of this mechanism and has to be addressed to enhance the confidence about classifier predications results.

Chapter 6

Conclusions and future of work

6.1 Conclusions

Despite the efforts invested by organisations to protect their resources against malicious activities, they are facing different types of attacks especially from trusted insiders who misuse their access privileges to admit illegitimate acts. One promising solution to this problem are self-adaptive systems , such as, the Self-Adaptive Authorisation Framework (SAAF). SAAF is capable of identifying internal attacks and responding to those attacks to mitigate and reduce their risk. However, such approaches have their own limitations , such as, insuring the accurate identification of known and unknown attacks. In addition, SAAF treats malicious behaviour at resource level and authorisation level independently. Hence, this project considers a machine learning approach to be a possible solution for those limitations and investigates its effectiveness through some experiments.

In order to investigate the effectiveness and the accuracy of machine learning approaches to capture anomalies in user behavior, some experiments have been conducted through four main phases:

- Phase 1: Classification using game application data to analyze detected abuses at the resource level.
- Phase 2: Classification using authorization level data to analyze the detected abuses at the system level.
- Phase 3: Classification using the combination of both application and authorization data to extend the detection scope and acquire better accuracy for the classifier, as both phases features were found complementing each other.
- Phase 4: Fusing game and authorization classifiers results to capture malicious behavior that is detected by one level classifier and has not been spotted by the other classifier.

It was valuable to evaluate the features and a combination of features (features selection phase) in order to identify which features or attributes are more

discriminative than others. This contributed eventually in refining classifier or detector performance through eliminating irrelevant and redundant features.

The contribution of this project is introducing a logical approach that positions machine learning to self-adaptive authorisation for the detection of malicious behavior. In conclusion, machine learning approach has been demonstrated to be a practical solution to merge with self-adaptive systems as a mean to extend the detection scope and enhance classification accuracy. Thus, a merged approach would be capable of identifying both known and possible unknown threats by targeting the fingerprints left by cybercriminals in a form of anomalous behaviour. Consequently, intrusion detection systems performance would be improved by considering both mechanisms, self adaptive systems and machine learning, for catching outliers. As machine learning tools are good at finding out the likelihood of new features sets to be malicious in context of other identified features.

Hence, it seems worthy for organizations to start looking for more advanced analysis tools and detection techniques oriented around machine learning concept to strive to reduce insider threats problem in access control systems.

Though, after analysing the experimental results, it was observed machine-learning tool does not produce 100% accuracy and assured results due to false positive predictions and levels of uncertainty. This could be considered as limitation of this mechanism and has to be tackled to enhance the confidence about classifier predications results. One the other hand, machine learning is a good technique that is continuously adjusting its baselines and learns its rules automatically from the context of training data and maps inputs to output. This benefit of machine learning is useful in reducing the time taken to modify and refine rules to detect new anomalies in user behavior. In addition, machine learning has been experienced here to be really fast in handling high amount of user behaviours data and producing the classification results in milliseconds. Moreover, it has been experienced that the accuracy of classifier can be boosted by providing a higher number of training data, as found with phase 2.

6.2 Future Work

What have been conducted throughout this project could be considered as a development stage for tackling insider threats within access control systems. The overall methodology could be improved for future work, by looking to the following improvement schemes:

- Use cross validation (testing method) to reduce the accuracy variance so that training and testing data can be utilized an equal number of times.
- Investigate accuracy improvement using "Bayesian networks" classifier instead of simple Naïve Bayes. In addition, examine how adding irrelevant attributes changes the classification accuracy of the Bayesian network model.
- Consider more features within the context of the game actions and the authorization levels and refines them, by excluding irrelevant once, so that overall certainty of a behavior to be abuse is improved. For example in the context of the game, introduce a new feature that capture session poisoning attacks where users reuse others authorizations to admit malicious behavior.
- Keep both authorizations and resources features in parallel and attempt to fuse their classifiers results using simple AND gate to increase the confidence on the classification results and reduce false positives.
- Introduce multiple classifiers to enhance the assurance of classification decision made.
- Utilize the features evaluator facility to classify the malicious behavior into different abuse classes according to the severity of attacks.

References

1. **Axelsson, S., & Sands, D. (2006).** An Introduction to Intrusion Detection. Understanding Intrusion Detection Through Visualization, 15-29.

2. **Bace, R., & Mell, P. (2001).** NIST special publication on intrusion detection systems. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

3. **Bailey, C. (2013).** Paramapada Sopaanam. [online] Saaf-resource.kent.ac.uk. Available at: https://saaf-resource.kent.ac.uk/game/game.php [Accessed 8 Sep. 2014].

4. **Bailey, C., Chadwick, D. W., & De Lemos, R. (2011, December).** Self-adaptive authorization framework for policy based RBAC/ABAC models. In Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on (pp. 37-44). IEEE.

5. **Bailey, C., Chadwick, D. W., & de Lemos, R. (2014).** Self-adaptive federated authorization infrastructures. Journal of Computer and System Sciences, 80(5), 935-952.

6. **Bertino, E., & Ghinita, G. (2011, March).** Towards mechanisms for detection and prevention of data exfiltration by insiders: keynote talk paper. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (pp. 10-19). ACM.

7. **Bertino, E., Terzi, E., Kamra, A., & Vakali, A. (2005, December).** Intrusion detection in RBAC-administered databases. In Computer security applications conference, 21st annual (pp. 10-pp). IEEE.

8. **Bishop, C. M. (2006).** Pattern recognition and machine learning (Vol. 1, p. 740). New York: springer.

9. **Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., & Scuse, D. (2013).** WEKA Manual for Version 3-7-8.

10. **Brancik, K., & Ghinita, G. (2011, February).** The optimization of situational awareness for insider threat detection. In Proceedings of the first ACM conference on Data and application security and privacy (pp. 231-236). ACM.

11. **Calders, T., & Verwer, S. (2010).** Three naive Bayes approaches for discrimination-free classification. Data Mining and Knowledge Discovery, 21(2), 277-292.

12. **Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., & Mitchell, T. M.** (2010, July). Toward an Architecture for Never-Ending Language Learning. In AAAI (Vol. 5, p. 3).

13. **Cert.org, (2001).** Insider Threat | The CERT Division. [online] Available at: http://www.cert.org/insider-threat/ [Accessed 6 Apr. 2014].

14. **Chadwick, D., Zhao, G., Otenko, S., Laborde, R., Su, L., & Nguyen, T. A. (2008).** PERMIS: a modular authorization infrastructure. Concurrency and Computation: Practice and Experience, 20(11), 1341-1357.

15. **Cheng, B. H., De Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., & Whittle, J. (2009).** Software engineering for self-adaptive systems: A research roadmap. In Software engineering for self-adaptive systems (pp. 1-26). Springer Berlin Heidelberg.

16. **Codeproject.com, (2012).** Naive Bayes Classifier - CodeProject. [online] Available at: http://www.codeproject.com/Articles/318126/Naive-Bayes-Classifier [Accessed 7 Sep. 2014].

17. **Cs.waikato.ac.nz, (2005).** Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [online] Available at: http://www.cs.waikato.ac.nz/~ml/weka/ [Accessed 15 Sep. 2014].

18. **Dodson, S. (2014).** Finding Unknown Threats with Anomaly Detection. [online] Information Security Buzz. Available at: http://www.informationsecuritybuzz.com/finding-unknown-threats-anomaly-detection/ [Accessed 4 Sep. 2014].

19. **Eberle, W., & Holder, L. (2009, April).** Graph-based approaches to insider threat detection. In Proceedings of the 5th annual workshop on cyber security and information intelligence research: cyber security and information intelligence challenges and strategies (p. 44). ACM.

20. **Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002).** A geometric framework for unsupervised anomaly detection. In Applications of data mining in computer security (pp. 77-101). Springer US.

21. **Fawcett, T. (2004).** ROC graphs: Notes and practical considerations for researchers. Machine learning, 31, 1-38.

22. **Giacomelli, P. (2013).** Apache Mahout Cookbook. Packt Publishing Ltd.

23. **Greitzer, F. L., Paulson, P., Kangas, L., Edgar, T., Zabriskie, M. M., Franklin, L., & Frincke, D. A. (2008).** Predictive modelling for insider threat mitigation. Pacific Northwest National Laboratory, Richland, WA, Tech. Rep. PNNL Technical Report PNNL-60737.

24. **Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009)**. The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.

25. **Hunker, J., & Probst, C. W. (2011).** Insiders and insider threats—an overview of definitions and mitigation techniques. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 2(1), 4-27.

26. **ID Analytics. (2008).** Analysis of internal data theft. White paper. Available at: http://www.idanalytics.com/media/Analysis-of-Internal-Data-Theft.pdf [Accessed 8 Sep. 2014].

27. **INCITS, A. (2004).** INCITS 359-2004, American national standard for information technology, role based access control. American National Standards Institute.

28. **Jasonb, (2014).** Classification Accuracy is Not Enough: More Performance Measures You Can Use | Machine Learning Mastery. [online] Available at: http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/ [Accessed 14 Aug. 2014].

29. **Jasonb (2014).** Feature Selection to Improve Accuracy and Decrease Training Time | Machine Learning Mastery. [online] Available at: http://machinelearningmastery.com/feature-selection-to-improve-accuracy-and-decrease-training-time/ [Accessed 11 Sep. 2014].

30. **Kantardzic, M. (2011).** Data mining: concepts, models, methods, and algorithms. John Wiley & Sons.

31. **Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998).** On combining classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(3), 226-239.

32. **Laskov, P., Düssel, P., Schäfer, C., & Rieck, K. (2005).** Learning intrusion detection: supervised or unsupervised?. In Image Analysis and Processing–ICIAP 2005 (pp. 50-57). Springer Berlin Heidelberg.

33. **Lane, T., & Brodley, C. E. (1997, February).** An application of machine learning to anomaly detection. In Proceedings of the 20th National Information Systems Security Conference (Vol. 377, pp. 366-380). Baltimore, USA.

34. **Lee, S. Y., Low, W. L., & Wong, P. Y. (2002).** Learning fingerprints for a database intrusion detection system. In Computer Security—ESORICS 2002 (pp. 264-279). Springer Berlin Heidelberg.

35. **Luminos, P. (2004).** paul.luminos.nl. [online] Paul.luminos.nl. Available at: http://paul.luminos.nl/document/198&comments [Accessed 19 Jun. 2014].

36. **Markov, Z., & Russell, I. (2006, June).** An introduction to the WEKA data mining system. In ACM SIGCSE Bulletin (Vol. 38, No. 3, pp. 367-368). ACM.

37. **Mathew, S., Petropoulos, M., Ngo, H. Q., & Upadhyaya, S. (2010, January).** A data-centric approach to insider attack detection in database systems. In Recent Advances in Intrusion Detection (pp. 382-401). Springer Berlin Heidelberg.

38. **McCormac, A., Parsons, K., & Butavicius, M. (2012)**. Preventing and Profiling Malicious Insider Attacks (No. DSTO-TR-2697). DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE DIV.

39. **Michalski, R. B., & Kubat, I. M (eds.) 1998.** Machine Learning and Data Mining, Methods and Applications.

40. **Mills, R. F., Peterson, G. L., & Grimaila, M. R. (2009)**. Insider Threat Prevention, Detection and Mitigation. In K. Knapp (Ed.), Cyber Security and Global Information Assurance: Threat Analysis and Response Solutions (pp. 48-74). Hershey, PA: Information Science Reference. doi:10.4018/978-1-60566-326-5.ch003

41. **Mitchell, T. M. (2005).** Logistic Regression. Machine learning, 10, 701.

42. **Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012).** Foundations of machine learning. MIT press.

43. **Mukkamala, S., Janoski, G., & Sung, A. (2002).** Intrusion detection using neural networks and support vector machines. In Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on (Vol. 2, pp. 1702-1707). IEEE.

44. **Murphy, K. P. (2012).** Machine learning: a probabilistic perspective. MIT press.

45. **Noon, E., & Li, H. (2010, May).** Defending against hit-and-run attackers in collaborative spectrum sensing of cognitive radio networks: A point system. In

Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st (pp. 1-5). IEEE.

46. **Nltk.org, (2014).** 6. Learning to Classify Text. [online] Available at: http://www.nltk.org/book/ch06.html [Accessed 14 Sep. 2014].

47. **Oltsik, J. (2014).** The 2013 Vormetric Insider Threat Report, ESG white paper. [online] Enterprise Strategy Group and Vormetric. Available at: http://www.vormetric.com/sites/default/files/vormetric-insider-threat-report-oct-2013.pdf [Accessed 2 Sep. 2014].

48. **Packtpub.com, (2013).** Implementing the Naïve Bayes classifier in Mahout | Packt. [online] Available at: https://www.packtpub.com/books/content/implementing-naïve-bayes-classifier-mahout [Accessed 7 Sep. 2014].

49. **Ruppert, B. (2009).** Protecting against insider attacks. SANS Institute.

50. **Sayad, S. (2012).** Naive Bayesian - Data Mining Map.. [online] Saedsayad.com. Available at: http://www.saedsayad.com/naive_bayesian.htm [Accessed 6 Sep. 2014].

51. **Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996)**. Role-based access control models. Computer, 29(2), 38-47.

52. **Sasaki, T. (2012).** A Framework for Detecting Insider Threats using Psychological Triggers. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 3(1/2), 99-119.

53. **Shin, D., & Claycomb, W. R. (2010)**. Detecting insider activity using enhanced directory virtualization (No. SAND2010-4655C). Sandia National Laboratories.

54. **Sommer, R., & Paxson, V. (2010, May).** Outside the closed world: On using machine learning for network intrusion detection. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 305-316). IEEE.

55. **Sundaram, A. (1996).** An introduction to intrusion detection. Crossroads, 2(4), 3-7.

56. **Sourceforge.net, (2014)**. Download Naive Bayes Classifier from SourceForge.net. [online] Available at: http://sourceforge.net/projects/naivebayesclass/files/NaiveBayesDemo.xls/download?use_mirror=garr&use_mirror=osdn [Accessed 19 Jun. 2014].

57. **Stehman, S. V. (1997).** Selecting and interpreting measures of thematic classification accuracy. Remote sensing of Environment, 62(1), 77-89.

58. **Stolfo, Salvatore, Bellovin, Steven M., Hershko and Shlomo**. (2008). Insider Attack and Cyber Security. New York, NY: Springer.

59. **Swets, J. A. (2014).** Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers. Psychology Press.

60. **Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009).** Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000.

61. **Vryniotis, V. (2014).** Developing a Naive Bayes Text Classifier in JAVA | Datumbox. [online] Blog.datumbox.com. Available at: http://blog.datumbox.com/developing-a-naive-bayes-text-classifier-in-java/ [Accessed 20 Jun. 2014].

62. **Witten, I. H., & Frank, E. (2005).** Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

63. **Witten, I. H., Frank, E., & Mark, A. (2011). Hall (2011).**" Data Mining: Practical machine learning tools and techniques.

64. **Yuan, E., & Tong, J. (2005, July).** Attributed based access control (ABAC) for web services. In Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on. IEEE.

Appendix I: Evaluation of Naïve Bayesian Tools

i. Implementing Naïve Bayes classifier in Mahout Unix based environment.

Mahout is basically Apache Software project that aims to produce free implementations of distributed and "scalable machine learning algorithms" and focuses on collaborative filtering, clustering and classification areas. The provided code contains Mahout ready scripts and other supplementary files that help to run Naïve Bayes algorithm on Unix command based environment. **(Packtpub.com, 2013).** Apache Mahout Cookbook provides a high range of coded examples that explain how to implement our own Bayesian tool **(Giacomelli, P., 2013)**.

Limitations: This tool is not flexible tool as it is fully command based and requires downloading all required utilities separately, which throws many errors while testing the package that require time to look back to them. In addition, it can only classify the instances without providing many analysis interoperations of results

ii. Free demonstration of Naïve Bayes algorithm based on the Visual Basic Language.

It classifies the attributes of a specific example like a weather conditions , such as, outlook, temperature, humidity and wind to test the applicability to play tennis or not. Based on the provided situation of each attribute, Naive Bayes will calculate the probability of each outcome and display the calculations on the screen, as illustrated in below figure **(Luminos, 2004)**:

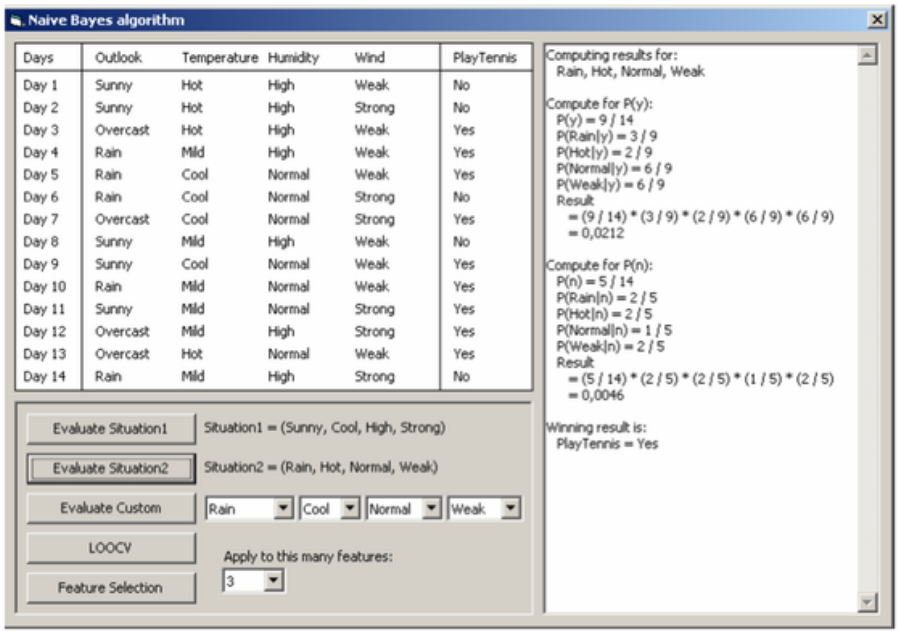


Figure II.1 Naive Bayes tool (weather example to play tennis or not)

Limitations: This tool provides executable version and source code. It is useful tool to understand the concept of Naive Bayes and how it is evaluating the situations besides providing some extra functionalities like feature selection. However, it is very limited on s specific example where it is difficult to run experiments without code modification. Furthermore, the performance measures of the tool cannot be visualized using this code.

iii. Open source codes in Java and C#, there are a variety of open source codes that implement Naïve Bayes Classifier in Java **(Vryniotis, 2014)** with a full access to classifiers, data objects, ready examples and attributes. In addition a couple of C# implementations provided by Codeproject.com. This code was tested on built in example that judge whether the person is female or male based on measured features (height, weight and foot size). **(Codeproject.com, 2012)**

Limitations: Such codes are very limited to predefined examples and require many code alterations to customize them to test the new experiments.

iv. Microsoft Visual Basic linked to MS Excel Sheets, this option provides a complete open source tool of an interactive Microsoft Excel Spreadsheet with Visual Basic implementation of Naïve Bayes. **(Sourceforge.net, 2014)**

Limitations: This tool could accept limited number on inputs and features, which limits its efficiency to process a high number on instances. In addition, it does not provide any level of analysis expect classification results.

v. Waikato Environment for Knowledge Analysis (WEKA), WEKA is java-based software **(Cs.waikato.ac.nz, 2005)** that contains a collection of machine learning algorithms that are designed to perform many data mining tasks. It is capable to develop the new machine-learning scheme **(Hall, M., et al, 2009)**. WEKA has been recognized as a "landmark system" over years across machine learning research communities and in data mining domain "because it is the only toolkit that has gained such widespread adoption and survived for an extended period of time" **(Markov, Z., & Russell, I., 2006)**. The algorithms can either be applied directly to a dataset or called from your own Java code. WEKA can provide both Graphical User Interface (GUI) and command line interface with a good range of built in functionalities , such as,: "data pre-processing, classification, regression, clustering, association rules, and visualization" Furthermore, it enables users to test and compare different machine learning algorithms on their data sets. **(Hall, M., et al, 2009)**.

Appendix II: Machine Learning Performance Measures

		Condition (as determined by "Gold standard")			
	Total population	Condition positive	Condition negative	Prevalence = Σ Condition positive / Σ Total population	
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV, Precision) = Σ True positive / Σ Test outcome positive	False discovery rate (FDR) = Σ False positive / Σ Test outcome positive
	Test outcome negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) = Σ False negative / Σ Test outcome negative	Negative predictive value (NPV) = Σ True negative / Σ Test outcome negative
	Positive likelihood ratio (**LR+**) = TPR/FPR	True positive rate (TPR, Sensitivity, Recall) = Σ True positive / Σ Condition positive	False positive rate (FPR, Fall-out) = Σ False positive / Σ Condition negative	Accuracy (ACC) = Σ True positive + Σ True negative / Σ Total population	
	Negative likelihood ratio (**LR−**) = FNR/TNR	False negative rate (FNR) = Σ False negative / Σ Condition positive	True negative rate (TNR, Specificity, SPC) = Σ True negative / Σ Condition negative		
	Diagnostic odds ratio (**DOR**) = LR+/LR−				

Image reference: http://en.wikipedia.org/wiki/Confusion_matrix

Appendix III : Experiments

Experiment No: 1

Objective:

- ✓ **The ultimate goal of this experiment was to understand how Naïve Bayes classifier behaves under different conditions represented by game features. Where, each feature represents a form of user attempts to attack the game by violating the game rules and policies.**

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **WEKA Data Mining Tool**

Procedure:

- **Initially, assume the existence of two features of game attacks and build manually meaningful patterns for the two attributes of the game, , such as,:**

1. **RD: Consecutive number of rolls of the dice.**
2. **MP: Number of places of moving the piece that exceeds the dice rolls value.**

So a crisp detector will detects if:

RD > 1 , to be classified as misuse.

RD > 3 , to be classified as abuse (Trigger adaptation)

MP >1 , to be classified as misuse.

MP >20 as abuse (Trigger adaptation)

- **This means that, by creating an independent a set of data that assumes that if the player moved beyond the roll dice values one place, the classifier is trained to detect that as misuse and if player moved by more than three places, it will be detected as abuse. In addition, whenever the player is rolling the dice consecutively more than 1 is misuse and if keeps rolling the dice more than 20 times, it should be classified as abuse.**

- **However, if two features have some different values but the combinations of them could leads to Trigger adaptation as well. For example: if RD=2 and MP =10, this could be classified as abuse as well, which is called two class problems. The detector should detects if there is abuse or not, based on a specific training dataset provided, in any supplied test dataset and classify them accordingly.**

- **The training set was prepared in a format that is compatible with our selected machine learning tool (WEKA), which is *.arff* format.**

Below is the two-attributes training dataset with its relevant class attributes:

```
@relation game
@attribute RD numeric
@attribute MP numeric
@attribute abuse {yes, no}
@data
4,21,yes
3,5,no
2,5,no
6,25,yes
2,4,no
1,5,no
2,3,no
4,30,yes
4,22,yes
5,40,yes
5,21,yes
1,1,no
2,6,no
1,2,no
0,1,no
1,0,no
5,26,yes
2,10,no
5,1,yes
6,3,yes
10,6,yes
1,21,yes
3,55,yes
1,1,no
3,20,no
4,21,yes
14,5,yes
4,19,yes
```

Results and Discussion

- By injecting the above training data set into WEKA, the following evaluation summery was shown:
  - Correctly Classified Instances 27 96.4286 %
  - Incorrectly Classified Instances 1 3.5714 %

Which means that, out of 28 provided training instances,27 of them are correctly classified with 96.4 % accuracy rate.

- By analyzing the predicted data from the classifier tool, it was noticed that once instance was not correctly classified as expected. The instance was exactly on the range of our classification, which was (3, 20). Which means that tool could not understand that aimed to exclude these two values from our classifications.

Then, Naive Bayes classifier was supplied by the following testing data set:

```
@data
3,5,no
4,18,yes
15,20,yes
3,23,yes
2,19,no
```

- After testing the training model by supplying the classifier with a few test data that aims to examine the misuse and abuse ranges and observe if the data were correctly classified by Naïve Bayes algorithm, the following output was obtained:
  - Correctly Classified Instances 4 80 %
  - Incorrectly Classified Instances 1 20 %

It indicates that one instance only was not correctly predicted as the actual supplied instance values. By looking deeply to the prediction summary, it was observed that, this instance was the last instance in the set (2, 19), that was very close to our classification ranges of abuse (3, 20), which concludes that, there is a level of uncertainty in Naïve Bayes classification because sometimes classifier is not certain exactly about our desired ranges and could leads to some false positive and true negative predictions.

- This step helps in visualizing the Naïve Bayes classification process and performance under some provided conditions. Moreover, an enhanced familiarity with machine learning tool was gained besides the understanding of data mining algorithm methodology in classifying instances and evaluating the test results.

Experiment No: 2

Objective:

- ✓ Extract a game feature (form of attack) that counts the number of places of moving the piece that exceeds the dice rolls value, where the player does not land on the right position.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs (It consists of: Audit table, Authentication table, Authorization table, Move table, Game table, Roll table, User table)

Procedure:

- Using Sequel Pro, create a new table to extract the feature information details and call it MP5.

```
-- Create a new table and call it MP5 to store the moving beyond dice roll feature results
CREATE TABLE `MP5` (
`roll_id` int(11) DEFAULT NULL,
`move_id` int(11) DEFAULT NULL,
`game_id` int(11) DEFAULT NULL,
`roll_value` int(11) DEFAULT NULL,
`start_pos` int(11) DEFAULT NULL,
`end_pos` int(11) DEFAULT NULL,
ILLEGALMOVE_Value int(11) DEFAULT NULL,
authz_id int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

- Write an SQL query that extracts the number of places of moving the piece beyond the dice roll value, which extract the information from *move* and *roll* tables.
    - The query is basically subtracting end position and starts position that player landed on, and taking care of the positions of ladders, snakes and bonus positions.
    - Then it joins the result with roll value, which is joined from roll table.
    - Then values are inserted in new table dedicated specially for this feature in order to make it easy to combine it with other features later.

```
-- Insert the results in the new created table
INSERT
INTO MP5 (roll_id,move_id, game_id, roll_value, start_pos, end_pos, ILLEGALMOVE_Value, authz_id)


SELECT move.roll_id, move.move_id, move.game_id, roll.value, move.start_pos, move.end_pos,
if (NOT (start_pos+value = 2 AND end_pos = 31)
AND  NOT (start_pos+ value = 8 AND end_pos = 24)
AND NOT  (start_pos+ value = 12 AND end_pos = 28)
AND  NOT  (start_pos+ value = 17 AND end_pos = 48)
AND NOT  (start_pos+ value = 27 AND end_pos = 54)
AND  NOT (start_pos+ value = 36 AND end_pos = 52)
AND   NOT (start_pos+ value = 40 AND end_pos = 57)
AND   NOT (start_pos+ value = 44 AND end_pos = 60)
AND   NOT (start_pos+ value = 20 AND end_pos = 4)
AND  NOT (start_pos+ value = 39 AND  end_pos = 10)
AND  NOT (start_pos+ value = 46 AND end_pos = 22)
AND  NOT (start_pos+ value = 51 AND end_pos = 33)
AND  NOT (start_pos+ value = 58 AND end_pos = 54)
AND  NOT (start_pos+ value = 63 AND end_pos = 47)
AND  NOT (start_pos+ value = 4 AND end_pos = 20)
AND  NOT (start_pos+ value = 10 AND end_pos = 39)
AND  NOT (start_pos+ value = 22 AND end_pos = 46)
AND   NOT (start_pos+ value = 33 AND end_pos = 51)
AND  NOT (start_pos+ value = 54 AND end_pos = 58)
AND  NOT (start_pos+ value = 47 AND end_pos = 63)
AND  NOT  (start_pos+ value = 7 AND end_pos = 49)
AND NOT   (start_pos+ value >= 64), (move.end_pos-move.start_pos)-roll.value, 0) as
ILLEGALMOVE_Value,move.authz_id  FROM move
left JOIN roll ON move.roll_id = roll.roll_id
```

Results & Discussion of Results:

- **If the user landed in the right position, the ILLEGALMOVE_Value will be zero, otherwise it will represent the number of illegal places the user moved to, or failed to move to.**
- **Number of illegal backward movements = 77 (negative values).**
- **Number of illegal forward movements = 4 (positive values), as can be seen below:**

roll_id	move_id	game_id	roll_value	move_beyond_dice
135	114	19	4	44
1087	677	144	5	43
1110	694	152	4	44
1196	742	161	2	61

- **The total number of misuses from this feature is 81.**
- **It was also observed that, there were 8 records where player moved without rolling the dice, as shown below:**

move_id	game_id	roll_id	turn	start_pos	end_pos	authz_id	last_updated
482	85	0	4	51	52	1499	2014-06-04 05:28:01
573	103	0	2	4	64	17	2014-06-13 12:27:57
574	103	0	3	4	64	1	2014-06-13 12:29:09
575	103	0	4	8	64	17	2014-06-13 12:30:25
581	110	0	1	1	64	18	2014-06-13 12:49:18
585	121	0	0	1	6	18	2014-06-13 13:20:31
594	125	0	1	1	62	19	2014-06-13 13:27:42
595	125	0	2	62	64	19	2014-06-13 13:28:32

This observation might be useful to consider as a feature as well.

Experiment No:3

Objective:

- ✓ Extract a game feature (a form of attack) that finds out if the user violates the game by of rolling the dice consecutively before actual move (RD).

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database

Procedure:

- Using Sequel Pro, create a new table to extract the feature information details and call it RD.

```
-- Create a new table and call it RD to store the consecutive rolls feature results


CREATE TABLE `RD`
(
  `game_id` int(11) DEFAULT NULL,
  `move_id` int(11) DEFAULT NULL,
  `roll_id` int(11) DEFAULT NULL,
  `authz_id` int(11) DEFAULT NULL,
  `rollsBeforeMove` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

- Write an SQL query that extracts the number of rolls of before making an actual move, which extract the information from *move* table.
    - The query is basically going through each roll id in the move table and calculates the difference between each two roll ids within the same game, so that if value calculated is more than one will give an indication about the existence of some consecutive rolls before an actual move of the player from his/her current position.
    - Then values are inserted in new table dedicated specially for this feature in order to make it easy to combine it with other features later.

```
INSERT INTO RD (game_id, move_id, roll_id, authz_id, rollsBeforeMove)
select m2.game_id,m2.move_id,m2.roll_id,m2.authz_id, if(m2.currentRollID is Null or m2.lastRollID is
Null or m2.currentRollID=0 or m2.lastRollID=0, 1, m2.currentRollID - m2.lastRollID) as rollsBeforeMove
from
(select
 m.move_id,
 m.game_id,
 m.roll_id,
 m.authz_id,
 m.roll_id as currentRollID,
 (
  select roll_id from move where move_id = (m.move_id-1)  and game_id - (m.game_id-1) = 1
 ) as lastRollID
from move as m
) as m2
```

Results & Discussion of Results:

- **If the user rolls the dice only one, Normal Case, the value of rollsBeforeMove will be equal to 1.**
- **If the user moved without rolling the dice, the value of rollsBeforeMove will be zero, and this case is considered as misuse as such move is illegal.**
- **The highest number of consecutive number of rolls before move is 14.**
- **The total number of misuses from this feature is 89.**
- **Writing the script of this feature was a bit challenging because sometimes, the roll id will be zero if there is a misuse of moving the piece without moving the dice. In addition, some games could be are played simultaneously which have to be considered.**

Experiment No: 4

Objective:

- ✓ Combine the two extracted features

3. RD: Consecutive number of rolling the dice before actual moves.
4. MP: Number of places of moving the piece that exceeds the dice rolls value.

into one table and group them based on *game level initially*.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database

Procedure:

- Using Sequel Pro, create a new table to store the extracted features information and call it test.

```
CREATE TABLE test
(
        game_id int,
        move_id int,
       ILLEGALMOVE_Value int,
       rollsBeforeMove int
)
```

- Write an SQL query that combines the extracted features into one table and select the other needed variables from other tables.

```
INSERT
INTO test (game_id, move_id, ILLEGALMOVE_Value, rollsBeforeMove)
SELECT MP5.game_id, MP5.move_id, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove
FROM MP5, RD
WHERE MP5.move_id = RD.move_id
```

- Export the resulting table as CSV (Comma Separated File) file with header fields, so that WEKA tool can accept the file for analysis.

Results & Discussion of Results:

- **Features new table is organized and order on game based, where features (attacks attempts) are ordered as per the game played.**
- **As can been seen from the table, very limited information are gained from each game were few number of violations could per observed at a time.**
- **User level analysis could be an improvement for this table.**

Experiment No:5

Objective:

- ✓ Combine the two extracted features

5. RD: Consecutive number of rolling the dice before actual moves.
6. MP: Number of places of moving the piece that exceeds the dice rolls value.

into one table and associate with them all the necessary variables to facilitate analysis on user level this time.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database

Procedure:

- Using Sequel Pro, create a new table to store the extracted features information and call it MP_and_RD.

```
CREATE TABLE MP_and_RD
(
        user_id varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
       ILLEGALMOVE_Value int,
       rollsBeforeMove int
)
```

- Write an SQL query that combines the extracted features into one table and select the other needed variables from other tables.

```
INSERT
INTO MP_and_RD (user_id,game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove)

SELECT  authorisations.user_id, MP5.game_id, MP5.move_id, MP5.roll_id, MP5.authz_id,
MP5.roll_value, authorisations.decision, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove

FROM MP5, RD,  authorisations
WHERE MP5.move_id = RD.move_id
AND authorisations.authz_id =  MP5.authz_id
ORDER BY authorisations.user_id, authorisations.authz_id
```

- Prior injecting the extracted two features in to Weka machine learning tool, train the two features such that if the user does not land on the right position with illegal move value not equal to 0, or if the user rolls the dice consecutively more than one, the classifier detects this behavior initially as abuse. In order to do this, first create a new column called “abuse” and then update its values as the following:

```
ALTER TABLE MP_and_RD
ADD abuse varchar (300)
UPDATE MP_and_RD SET abuse = if(ILLEGALMOVE_Value!=0 or rollsBeforeMove!=1, "YES", "NO")
```

- Export the resulting table as CSV (Comma Separated File) file with header fields, so that WEKA tool can accept the file for analysis.

Results & Discussion of Results:

- Features new table is organized and order on user based, where features (attacks attempts) can be analyzed and related to each user.
- Game level analysis is useless as it is limited to very limited results and information of attacks.
- User level analysis will reveal each user attempts to hack the game, as features results could be accumulated per user.

Experiment No: 6

Objective:

- ✓ Preprocess the first two application features (game attacks) using Weka machine learning tool.
- ✓ Classify the selected features using Naïve Bayes Algorithm.

Tools and Data used:

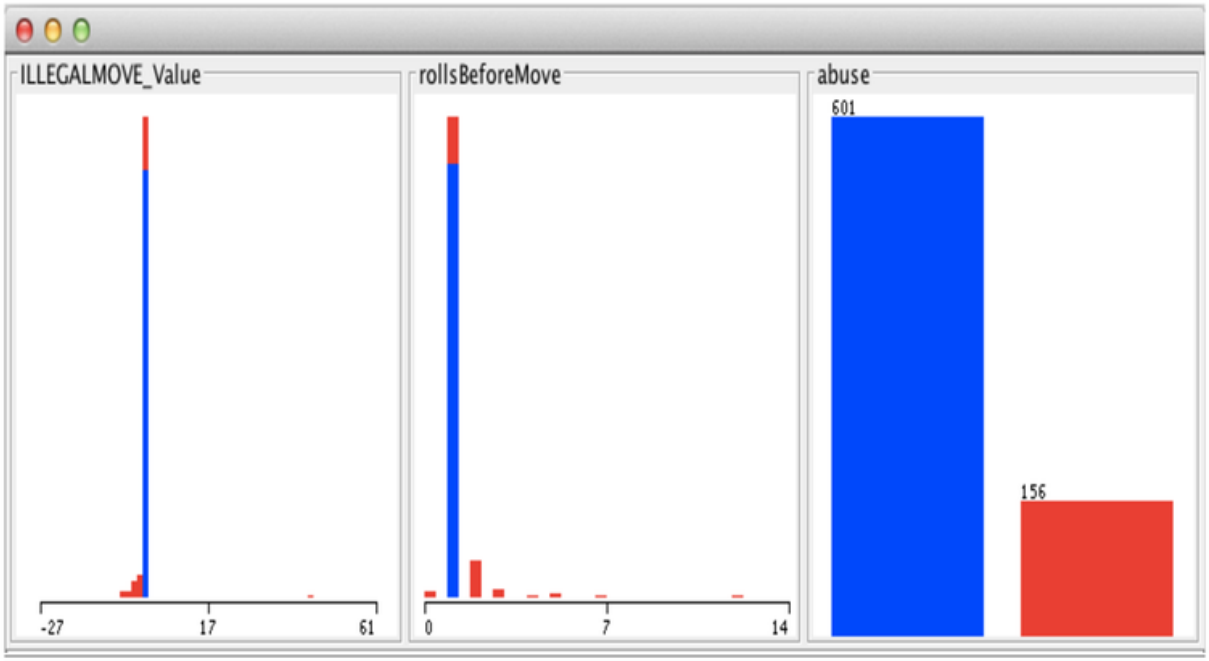
- WEKA machine learning tool. (Waikato Environment for Knowledge Analysis, Version 3.6.11)
- The data file used is called: MP_and_RD.csv

Procedure:

- Inject the extracted two features, which are stored in (MP_and_RD.csv) file into Weka tool, by following below steps.
  - Open Weka machine learning tool > Explorer > Preprocess
  - Open file: open the saved CSV file
  - In the preprocess tab: we can visualize the attributes and take a look for the following information for each attribute: (Minimum, Maximum, Mean and Standard Deviation). In addition, the number of abuses can be extracted from Class Visualize tab.
  - In order to classify instances using Naïve Bayes algorithm, click on Classify tab > choose > weka > classifiers > bayes > Naïve Bayes
  - In Test Options: we can classify the training set, supply test data set, use Cross Validation folds, or spilt the full data set into training and test data using: Percentage Split Option.
  - For this experiment, the full dataset has been spilt into 70 % for training data and 30 % for the test data.
  - In order to show the classifier predictions vs. actual predictions, select: more options> tick : output predictions
  - The to start the classification just simply click on : Start
- Then observe the classifier output, which provides the predictions on test split, Evaluation on test split, Detailed Accuracy By Class and shows the Confusion Matrix results.

Results & Discussion of Results:

Pre-process data results:



The full data file indicates that, 156 are determined to be abuse whereas 601 instances are non-abuse. The total number of instances is 757.

Naive Bayes Classifier output:

Class

Attribute	NO	YES
	(0.79)	(0.21)

==========

ILLEGALMOVE_Value

mean	0	-0.141
std. dev.	1.2222	8.7273
weight sum	601	156
precision	7.3333	7.3333

rollsBeforeMove

mean	1.1667	2.3408
std. dev.	0.1944	2.2085
weight sum	601	156
precision	1.1667	1.1667

- === Evaluation on test split ===

  === Summary ===

  - Correctly Classified Instances 192 84.5815 %
  - Incorrectly Classified Instances 35 15.4185 %
  - Kappa statistic 0.4408
  - Mean absolute error 0.1716
  - Root mean squared error 0.3841
  - Relative absolute error 50.8224 %
  - Root relative squared error 90.4363 %
  - Total Number of Instances 227

- === Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.66	0.833	1	0.909	0.67	NO
	0.34	0	1	0.34	0.507	0.67	YES
Weighted Avg.	0.846	0.506	0.872	0.846	0.815	0.67	

Discussion of results:

- The first feature that represents distance moved beyond the roll dice value have much higher precession (7.3333) than the second feature that count the number of consecutive rolls thrown before an actual move, which comes with only 1.1667 precision.
- Out of 757 provided instances from game log, 227 instances (30 % of the dataset) have been used for testing as per our setting of the dataset spilt were the rest of the data have been used to train the classifier which forms remaining 70 % of the dataset.
- Out of 227 test instances, 192 were correctly classified with 84.5815 % accuracy rate and naïve bays classifier incorrectly classified the remaining 35. After comparing the actual classification versus tool classification, it was observed that whenever the data is close to my pre-defined thresholds (at the threshold, greater than and less that by small amount) the classifier is sometimes record some of them as misclassifications.
- The detailed Accuracy By Class reflects the performance measures of the classifier, and in this experiment, the following have been observed:
  - TP Rate (TPR): rate of true positives is equal to 1, which reflect that all instances correctly classified as for "NO" class and it is 0.34 for YES class because only 32 instances were truly classified as "YES".
  - FP Rate (TFR): rate of false positives is equal to 0.66 because of the 35 instances that were

falsely classified as "YES" and it is 0 as there is no instances falsely classified as YES.

- **Precision:** "NO" class has 0.833 precision, whereas "YES" class achieved 1 precision because of the accurate classification of all instances as YES.
- **Recall:** "NO" class has 1 recall, which directly calculated by dividing the number of True Positives (174) by the number of True Positives (174) and the number of False Negatives (0). Whereas, "YES" class achieved 0.34 recall which is calculated in similar way to NO class.
- **F-Measure:** 0.909 for the combined measure for precision and recall of No class and 0.507 for YES class.
- **ROC:** The area under the curve is equal to 0.67 for both YES and NO classes.
- **Confusion matrix** represents that:
  - 174 true positives (actual non-abuse that were correctly classified as NO)
  - 35 false positives (abuse that were incorrectly labeled as Non-abuse)
  - 0 false negatives (Non-abuse that were incorrectly marked as abuse)
  - 18 true negatives (all the remaining instances, correctly classified abuse YES)

➢ In order to enhance the accuracy of the classifier, it would be better to combine the above-discussed features with other related features and re-classify the attributes with more features.

Experiment No: 7

Objective:

- Extract a new feature from the game that calculates the accumulative value of the number illegal places the user moved to, or failed to move to.
- Extract a new feature from the game that calculates the accumulative value of the number of consecutive number of rolls before the player moves.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- The output table (MP_and_RD) obtained from the last experiment (5) will be used for this experiment.

Procedure:

- Using Sequel Pro, create a new table to extract the new feature information details and call it feature.

```
CREATE TABLE feature
(
        id int not NULL AUTO_INCREMENT,
        game_id int,
        move_id int,
       ILLEGALMOVE_Value int,
       rollsBeforeMove int,
      CUM_MP int, -- to store the accumulative value of MP
     CUM_RD int, -- to store the accumulative value of RD
      PRIMARY KEY (id)
)
```

- Then, insert into the newly created feature table both ILLEGAL MOVE Value and rolls Before Move value which are already extracted from the last experiment and leave the accumulative values fields as zeros and ready to be updated.

```
INSERT
INTO feature (user_id,game_id, move_id,ILLEGALMOVE_Value, rollsBeforeMove)

SELECT  game.user_id, MP5.game_id, MP5.move_id, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove

FROM MP5, RD, game
WHERE MP5.move_id = RD.move_id
AND game.game_id = MP5.game_id
```

- **Test how to run a counter into SQL, many SQL queries and scripts were tried first, for example:**

```
DECLARE @counter INT
SET @counter = 0
WHILE @counter <>
  BEGIN
    SET @counter = @counter + 1
    PRINT 'The counter : ' + CAST(@counter AS CHAR)
  END DECLARE @counter INT
SET @counter = 0
WHILE @counter <>
  BEGIN
    SET @counter = @counter + 1
    PRINT 'The counter : ' + CAST(@counter AS CHAR)
  END
```

AND:

```
create procedure update()
begin
declare intCounter int;
set intCounter = 0;
loop
update intCounter = intCounter + 1;
where ILLEGALMOVE_Value !=0;
end loop;
end
```

- It was observed that: 28 rows affected if two features applied.
- Later, write an SQL script in a form of (procedure update) that extracts the cumulative number of places of moving the piece beyond the dice roll value, which extract the information from *MP5* and *RD recently created* tables.
  - The SQL script is basically should be running a procedure that have a loop with a counter to check the current and previous records and check if the user does not land on the right position, or if the user rolled the dice more than once before move, update the cumulative value by the absolute value of *ILLEGALMOVE_Value* and *rollsBeforeMove*
  - In order to check if the updating procedure is working properly, just customize it first to update the *ILLEGALMOVE_Value accumulative value which is called* updateCMP()PROCEDURE, and the code is as follows:

```
DELIMITER $$
CREATE PROCEDURE updateCMP()
BEGIN
DECLARE v_mp, v_user_id, v_move_id, v_id INT;
DECLARE v_last_move_id INT;
DECLARE cmp INT;
DECLARE done INT DEFAULT FALSE;
DECLARE curl CURSOR FOR SELECT id, user_id, move_id, ILLEGALMOVE_Value FROM featureByUser;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
        OPEN curl;
        the_loop: LOOP

                FETCH curl INTO v_id, v_user_id, v_move_id, v_mp;

                IF done THEN
                        LEAVE the_loop;
                END IF;

                SELECT move.move_id INTO v_last_move_id from move, game
                        WHERE move_id < v_move_id
                        AND game.user_id = v_user_id
                        AND move.game_id = game.game_id
                        ORDER BY move_id DESC LIMIT 1;
```

```
            IF length(v_last_move_id = 0) THEN
                SET cmp = v_mp;
            ELSE
                SELECT featureByUser.CUM_MP INTO cmp FROM featureByUser
                WHERE featureByUser.move_id = v_last_move_id;
            END IF;

            UPDATE featureByUser SET CUM_MP = cmp WHERE move_id = v_move_id;

        END LOOP the_loop;
        CLOSE curl;
END$$
DELIMITER ;
```

- **Check the status of the procedure using the following command:**

  show procedure status

Results & Discussion of Results:

- **Unfortunately, after spending a huge time and efforts to update the accumulative values of both MP and RD features, the loop procedure update does not work properly. This could be due to the failure of the loop to check all features records as desired and update them probably.**
- **Because of the time constraint, the accumulative values could be calculated using Excel Spread Sheet based on MP_and_RD table, and export the results as CSV file for Weka machine learning analysis.**

Experiment No: 8

Objective:

Re-do experiment 6 using different approach to:

- Extract a new feature from the game that calculates the accumulative value of the number illegal places the user moved to, or failed to move to.
- Extract a new feature from the game that calculates the accumulative value of the number of consecutive number of rolls before the player moves.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- Microsoft Excel Spread Sheet.
- Weka tool.

Procedure:

- Using Sequel Pro, create a new table to extract the new feature information details and call it featureTable.

```
CREATE TABLE featuresTable
(
        id int not NULL AUTO_INCREMENT,
        user_id varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
ILLEGALMOVE_Value int,
rollsBeforeMove int,
MP_misuse int,
RD_misuse int,
misuse_count_per_move  int,
```

```
CUM_MP int,
CUM_RD int,
PRIMARY KEY (id)
)
```

- Then, insert into the newly created feature table both ILLEGAL MOVE Value and rolls Before Move value which are already extracted from the last experiment and associate them with their relevant: user id, game id, move id, roll id, authorization id, roll value and decision.

```
INSERT
INTO featuresTable (user_id,game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)

SELECT  authorisations.user_id, MP5.game_id, MP5.move_id, MP5.roll_id, MP5.authz_id,
MP5.roll_value, authorisations.decision, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove,
if(MP5.ILLEGALMOVE_Value!=0, 1, 0 ) as MP_misuse, if(RD.rollsBeforeMove!=1, 1, 0) as RD_misuse, 0
as misuse_count_per_move

FROM MP5, RD, game, authorisations
WHERE MP5.move_id = RD.move_id
AND game.game_id = MP5.game_id

AND authorisations.authz_id =  MP5.authz_id
ORDER BY game.user_id
```

```
-- in order to facilitate the calculation for the accumulative values, set the values
--whenever aattack happened due those features as misuse
UPDATE featuresTable SET misuse_count_per_move = MP_misuse + RD_misuse
```

- Then, export the featureTable as an excel file to perform the accumulative values calculations.
- The accumulative values have been calculated by incrementing the value starting from zero, whenever there is misuse by corresponding absolute values of MP and RD.
- Prior injecting the extracted two features in to Weka machine learning tool, train the data composed of the four features such that tool should detects abuse in the following conditions:

- If the user does not land on the right position with illegal move value not equal to 0.
- If the user rolls the dice consecutively more than one.
- If the accumulative value of MP is greater than 21.
- If the accumulative value of RD is greater than or equals to 10.

- Export the resulting table into a CSV file, and inject the data file into weka tool.
- Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.

Results & Discussion of Results:

- The number of instances in the full data set is 748.

➢ === Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	204	91.0714 %
Incorrectly Classified Instances	20	8.9286 %
Kappa statistic	0.7475	
Mean absolute error	0.1085	
Root mean squared error	0.2941	
Relative absolute error	29.3295 %	
Root relative squared error	66.9856 %	
Total Number of Instances	224	

➢ === Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.982	0.293	0.906	0.982	0.942	0.884	NO
	0.707	0.018	0.932	0.707	0.804	0.884	YES
Weighted Avg.	0.911	0.222	0.912	0.911	0.906	0.884	

=== Confusion Matrix ===

a	b	<-- classified as
163	3	a = NO
17	41	b = YES

- **The results reveals that there are:**
  - **163 true positives (actual non-abuse that were correctly classified as NO)**
  - **17 false positives (abuse that were incorrectly marked as Non-abuse)**
  - **3 false negatives (Non-abuse that were incorrectly marked as abuse)**
  - **41 true negatives (all the remaining instances, correctly classified abuse YES)**
  - **High recall indicates that there are less false negatives.**

- **The classifier model output indicates that ILLEGALMOVE_Value feature has the higher precision with (7.333) followed by its accumulative value feature (3.625). Whereas other features came with much less precision , such as, 2 for accumulative value feature for RD and just 1.7692 for RD feature.**

- **Important modification required:**
  - **There are 9 instances are missing from the move table are not combined with other data.**
  - **The total number of instances as per move table should be 757.**

Figure 1: Classifier model output:

Classifier output

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                          Class
Attribute                   NO      YES
                         (0.76)  (0.24)
=================================
ILLEGALMOVE_Value
  mean                        0  0.1236
  std. dev.              1.2222  8.1703
  weight sum                570     178
  precision              7.3333  7.3333

rollsBeforeMove
  mean                   1.7692  2.5246
  std. dev.              0.2949  2.5561
  weight sum                570     178
  precision              1.7692  1.7692

CUM_MP
  mean                   2.0033  7.6369
  std. dev.              2.9415  8.1547
  weight sum                570     178
  precision               3.625   3.625

CUM_ILLEGAL_RD
  mean                   1.2105  5.6742
  std. dev.              2.1976  7.5316
  weight sum                570     178
  precision                   2       2
```

Experiment No: 9

Objective:

- The ultimate goal of this experiment is combine the all extracted features with the correct number of instances (757) as per move table.
- Re-do experiment 7 and:

1. Extract a new feature from the game that calculates the accumulative value of the number illegal places the user moved to, or failed to move to.
2. Extract a new feature from the game that calculates the accumulative value of the number of consecutive number of rolls before the player moves.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- Microsoft Excel Spread Sheet.
- Weka Tool.

Procedure:

- Using Sequel Pro, create a new table to extract the new feature information details and call it feature Table.

```
CREATE TABLE featuresTable
(
        id int not NULL AUTO_INCREMENT,
        user_id varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
       ILLEGALMOVE_Value int,
       rollsBeforeMove int,
      MP_misuse int, -- to be set if ILLEGALMOVE_Value not zero
      RD_misuse int,  -- to be set if  rollsBeforeMove not equals to one
```

```
    misuse_count_per_move  int,
     CUM_MP int,
    CUM_RD int,
    PRIMARY KEY (id)
)
```

Modify the SQL query such that it just do the selection from MP5, RD and authorisations tables and only if they have the same move_id and authorization id.

```
MP5.move_id = RD.move_id
authorisations.authz_id =  MP5.authz_id
```

- Insert the selected values in to the created featuresTable and associate them with their relevant information: user id, game id, move id, roll id, authorization id, roll value and decision.

```
INSERT
INTO featuresTable (user_id,game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)

SELECT  authorisations.user_id, MP5.game_id, MP5.move_id, MP5.roll_id, MP5.authz_id,
MP5.roll_value, authorisations.decision, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove,
if(MP5.ILLEGALMOVE_Value!=0, 1, 0 ) as MP_misuse, if(RD.rollsBeforeMove!=1, 1, 0) as RD_misuse, 0
as misuse_count_per_move

FROM MP5, RD,  authorisations
WHERE MP5.move_id = RD.move_id
AND authorisations.authz_id =  MP5.authz_id
ORDER BY authorisations.user_id, authorisations.authz_id
```

```
-- in order to facilitate the calculation for the accumulative values, set the values
--whenever aattack happened due those features as misuse

UPDATE featuresTable SET misuse_count_per_move = MP_misuse + RD_misuse
```

- Then, export the featureTable as an excel file to perform the accumulative values calculations.
- The accumulative values have been calculated by incrementing the value starting from zero, whenever there is misuse by corresponding absolute values of MP and RD.
- Prior injecting the extracted two features in to Weka machine learning tool, train the data composed of the four features such that tool should detects abuse in the following conditions:
    - If the user does not land on the right position with illegal move value not equal to 0.
    - If the user rolls the dice consecutively more than one.
    - If the accumulative value of MP is greater than 21.
    - If the accumulative value of RD is greater than or equals to 10.
- Export the resulting table into a CSV file, and inject the data file into weka tool.
- Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.

Results & Discussion of Results:

- Classifier output:

➢ === Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	195	85.9031 %
Incorrectly Classified Instances	32	14.0969 %
Kappa statistic	0.6007	
Mean absolute error	0.1466	
Root mean squared error	0.3658	
Relative absolute error	38.6083 %	
Root relative squared error	81.7958 %	
Total Number of Instances	227	

➢ === Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.969	0.453	0.845	0.969	0.903	0.841	NO
0.547	0.031	0.875	0.547	0.673	0.841	YES

Weighted Avg. 0.85 0.334 0.853 0.85 0.838 0.841

- === Confusion Matrix ===

  a b <-- classified as

  158 5 | a = NO

  29 35 | b = YES

- The results reveals that there are:
  - 158 true positives (actual non-abuse that were correctly classified as NO)
  - 5 false positives (abuse that were incorrectly marked as Non-abuse)
  - 29 false negatives (Non-abuse that were incorrectly marked as abuse)
  - 35 true negatives (all the remaining instances, correctly classified abuse YES)
  - High recall indicates that there are less false negatives.
- The classifier model output indicates that ILLEGALMOVE_Value feature has the higher precision with (7.333) followed by its accumulative value feature (2.7737). Whereas other features came with much less precision and nearly the same precision (1.8696 for accumulative value feature for RD and just 1.7692 for RD feature).
- The extracted four features needs to be evaluated to see the most relevant and important features that affect classifier performance.
- The extracted four features needs to filtered so that powerful features selected for the next phase.

Experiment No: 10

Objective:

- Classify the accumulative values features only, that are extracted in the last experiment:
  - If the accumulative value of MP (moving the dice beyond the roll value)
  - If the accumulative value of RD (rolling the dice consecutively before move)

Tools and Data used:

- Weka tool.
- Features injected in the tool:
  1. If the accumulative value of MP is greater than 21.
  2. If the accumulative value of RD is greater than or equals to 10.

  Which are available in (step1 accum features.csv) data file.

Procedure:

- Pre-process the data file: step1 accum features.csv
- Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.

Results & Discussion of Results:

➢ === Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	217	95.5947 %
Incorrectly Classified Instances	10	4.4053 %
Kappa statistic	0.8032	
Mean absolute error	0.0442	
Root mean squared error	0.1752	
Relative absolute error	24.3669 %	
Root relative squared error	56.958 %	
Total Number of Instances	227	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.951	0	1	0.951	0.975	0.996	NO
	1	0.049	0.706	1	0.828	0.996	YES
Weighted Avg.	0.956	0.005	0.969	0.956	0.959	0.996	

=== Confusion Matrix ===

```
  a   b   <-- classified as
193  10 |  a = NO
  0  24 |  b = YES
```

As can be seen from above results:

1. **The accuracy of the classifier has been improved compared to the last experiment by approximately 10%**
2. **Only 10 instances have been misclassified leading to 10 false negatives.**
3. **ROC area has been improved drastically as a result of the high rate of true positive.**
4. **The problem of misclassification is only limited to "NO" class.**
5. **False positive rate is really good in this experiment with optimal value (1) as there is no instances were misclassified as false positive.**

- **Since the accumulative values features are originated from the original RD and MP features, they could give good indications about them in their absence.**

Experiment No: 11

Objective:

- Evaluate all the attributes (features) from the application level of the game.

Tools and Data used:

- Weka tool.
- Features to be evaluated:
  3. If the user does not land on the right position with illegal move value not equal to 0.
  4. If the user rolls the dice consecutively more than one.
  5. If the accumulative value of MP is greater than 21.
  6. If the accumulative value of RD is greater than or equals to 10.

  Which are available under in (step1 features.csv) data file.

Procedure:

- Use (Select attributes) tab provided by Weka tool in order to measure the worth of each attribute in our Naïve Bayes Classification.
- Select InfoGain (Information Gain) as attribute evaluator for the data file.
- Select Ranker method in order to generate the attributes rankings as per their importance.

Results & Discussion of Results:

➢ Evaluation mode: evaluate on all training data

```
=== Attribute Selection on all input data ===
Search Method:
        Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 5 Abuse):
        Information Gain Ranking Filter
Ranked attributes:
 0.252  2 rollsBeforeMove
 0.248  1 ILLEGALMOVE_Value
 0.215  4 CUM_RD
 0.141  3 CUM_MP
Selected attributes: 2,1,4,3 : 4
```

- **InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class. InfoGain(Class,Attribute) = H(Class) – H(Class | Attribute), where H is the information entropy.[2]**
- **It evaluates our attributes as the following:**
    1. **The most important attribute that has a higher ranking is the consecutive number of rolls before move (RD).**
    2. **The least powerful attribute that came on last rank is the accumulative value of MP.**

[2] http://www.opentox.org/dev/documentation/components/infogainattribeval

Experiment No: 12

Objective:

Work on authorization log and extract some features like:

1. Deny Count : that represents if the user exceeds a threshold limit of Denies, it will be classified as misuse
2. Count Number of Consecutive Actions: if any action is repeated consecutively and exceeds a certain limit, it will be classified as misuse
3. Count number of Unknown Actions: if the number of unknown actions exceeds a certain limit, it will be classified as misuse

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- Microsoft Excel Spread Sheet.
- Weka Tool.

Procedure:

- Using Sequel Pro, create a new table that contains the necessary information required of features extraction.

```
CREATE TABLE authFeatures(
user_id VARCHAR( 300 ) ,
action VARCHAR( 300 ) ,
authz_id INT,
decision VARCHAR( 300 ) ,
Set_if_Deny INT,
PRIMARY KEY ( authz_id )
)
```

- Set_if_Deny field is used to facilitate the calculation where this field will be set to one, whenever access request is denied.
- Insert the selected values in to the created authFeatures and associate them with their relevant information.

```
INSERT
INTO authFeatures (user_id, authz_id,action, decision, Set_if_Deny)

SELECT user_id, authz_id,action, decision, if(decision="deny", 1, 0) as Set_if_Deny
FROM  authorisations
ORDER BY user_id, authz_id
```

- Then, export the authFeatures as an excel file to perform the features calculations.
- The features are calculated as the following:
  - If deny is set, Deny count will accumulate the value after checking that they belongs to the same user.
  - Count number of consecutive actions if the current action is the same as the previous actions, the counter will be incremented after checking that they belongs to the same user.
  - Count number of unknown action: the counter is incremented by one whenever the action is not in the allowed actions after checking that they belongs to the same user. If the user is different, the counter will be restarted from zero. The statement should be something similar to:

IF(current user =previous user,IF(OR(action="start", action ="bonus", action ="roll", action ="move", action ="ladder", action ="end"), Count Uknown Actions, Count Uknown Actions +1),IF(OR(action="start", action ="bonus", action ="roll", action ="move", action ="ladder", action ="end"),0,0+1))

- Prior injecting the extracted two features in to Weka machine learning tool, train the data composed of the three features such that tool should detects abuse in the following conditions:
  - *If Deny Count is greater than or equal to 5, OR*
  - *If Count Number of Consecutive Actions is greater than or equal to 10, OR*
  - *If the Countof Uknown Actions is greater than or equal to 3*
- Export the resulting table into a CSV file (final authorisation features only) , and inject the data file into weka tool.
- Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.
- Evaluate the selected three attributes using InfoGain and Ranker method, to evaluate the worth of

each attribute in the classification process.

Results & Discussion of Results:

- Classifier output:
- === Evaluation on test split ===

  === Summary ===
- Correctly Classified Instances 699 95.8848 %
- Incorrectly Classified Instances 30 4.1152 %
- Kappa statistic 0.8523
- Mean absolute error 0.0547
- Root mean squared error 0.2017
- Relative absolute error 19.4578 %
- Root relative squared error 53.8479 %
- Total Number of Instances 729

- === Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.977	0.13	0.974	0.977	0.975	0.988	NO
	0.87	0.023	0.884	0.87	0.877	0.988	YES
Weighted Avg.	0.959	0.112	0.959	0.959	0.959	0.988	

- === Confusion Matrix ===

  a b <-- classified as

  592 14 | a = NO

  16 107 | b = YES
- The results reveals that there are:
- 592 true positives (actual non-abuse that were correctly classified as NO)
- 16 false positives (abuse that were incorrectly marked as Non-abuse)
- 14 false negatives (Non-abuse that were incorrectly marked as abuse)
- 107 true negatives (all the remaining instances, correctly classified abuse YES)

- High recall indicates that there are less false negatives.
- With reference to classifier model output, the classifier model output indicates the precision of all the three feature is equals to 1 for both classes (YES and NO)
- Other features could extracted from authorisation log , such as,: rate of rolls, rate of moves, rate of starts..etc in a given time interval.
- According the attributes evaluation:
  1. Count Number of Consecutive Actions scored the first rank with approximately 0.474 contribution.
  2. Deny Count scored the second rank with approximately 0.182 % contribution.
  3. Count number of Unknown Actions scored the third and last rank with approximately 0.089 % contribution.

     Which reflects the high importance of the Number of Consecutive Actions counter in the classification of authorisation features.

Figure 1: Classifier model output

Classifier output

```
=== Classifier model (full training set) ===

Naive Bayes Classifier

                                          Class
Attribute                                   NO      YES
                                         (0.83)  (0.17)
============================================================
Deny Count
  mean                                   0.2551  4.9173
  std. dev.                              0.7346  7.6739
  weight sum                               2019     411
  precision                                   1       1

Count Number of Consecutive  Actions
  mean                                   1.9946 15.2457
  std. dev.                              2.4777 10.2033
  weight sum                               2019     411
  precision                                   1       1

Count Uknown Actions
  mean                                   0.0263  2.8783
  std. dev.                              0.1775  6.4332
  weight sum                               2019     411
  precision                                   1       1
```

Figure 2: attributes evaluations:

Attribute selection output

```
=== Run information ===

Evaluator:    weka.attributeSelection.InfoGainAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     final authorisation features only
Instances:    2430
Attributes:   4
              Deny Count
              Count Number of Consecutive  Actions
              Count Uknown Actions
              Abuse
Evaluation mode:evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 4 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.4748  2 Count Number of Consecutive  Actions
 0.182   1 Deny Count
 0.0896  3 Count Uknown Actions

Selected attributes: 2,1,3 : 3
```

Experiment No: 13

Objective:

Combine both authorization log and game application log features into one dataset file and classify them with Naïve Bayes algorithm as one dataset, which are:

1. Deny Count : that represents if the user exceeds a threshold limit of Denies (5), it will be classified as misuse
2. Count Number of Consecutive Actions: if any action is repeated consecutively and exceeds a certain limit (10), it will be classified as misuse
3. Count number of Unknown Actions: if the number of unknown actions exceeds a certain limit (3), it will be classified as misuse
4. RD: Consecutive number of rolling the dice before actual moves.
5. MP: Number of places of moving the piece that exceeds the dice rolls value.
6. The accumulative value of the number illegal places the user moved to, or failed to move to.
7. The accumulative value of the number of consecutive number of rolls before the player moves.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- Microsoft Excel Spread Sheet.
- Weka Tool.

Procedure:

- Write an SQL query statement that select and combine the required information for both application level features and authorisation features based on the common authz_id.

```
SELECT authorisations.user_id,authorisations.action, featuresTable.game_id,
featuresTable.move_id,featuresTable.roll_id,
authorisations.authz_id, featuresTable.roll_value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
LEFT OUTER JOIN featuresTable
```

```
ON featuresTable.`authz_id`=authorisations.`authz_id`
ORDER BY authorisations.user_id, authorisations.authz_id, featuresTable.move_id
```

Results & Discussion of Results:

- While trying to combine the application features and authorisation features, I got a problem with some duplicated authz_id in the move table where with the same authz_id, the player is having sometimes different move_id, game_id and roll_id.
- The number of records in authorisation table are 2430, however if we combine both game and authorization data, the total number of instance increased to 2443 with 13 additional instance.
- In order to find the duplicates in SQL:

```
SELECT
  authz_id, move_id, game_id, roll_id, COUNT(*)
FROM
  move
GROUP BY
  authz_id
HAVING
  COUNT(*) > 1
```

- There are four duplicated authz_ids (which are: 17, 18, 19, 2098) as shown below with the count of their occurrence:

authz_id	move_id	game_id	roll_id	COUNT(*)
17	573	103	0	2
18	8	1	8	3
19	594	125	0	2
2098	649	140	1052	10

As can be seen from the above table, there are:

1 additional record for authz_id = 17

2 additional record for authz_id = 18

1 additional record for authz_id = 19

9 additional record for authz_id = 2098

With a total of 13 records as expected.

- ***Possible explanation By Chris:***

The likelihood is that the player carried out a session poisoning attack.

For example, take move_id 581.

The player has reused an old authz_id to for an old move 109 games prior!

In session poisoning, a user can inject certain session values which are used when logging. The important records are generated through the player playing the game, and are never stored in a session, but in the case of the game logging authorisation, it considers 2 steps: log authorisation request, log authorised move. Because the authorisation request is logged previous to the actual move, the attacker is putting in their own value (basically overruling the need for authorisation) and exploiting vulnerability in the game.

- **By going deep to analyze some of duplicated authorization id's after the Concatenation process:**

id	user_id	action	game_id	move_id	roll_id	authz_id	roll_value	decision
1223	mike	roll	103	573	0	17	0	permit
1224	mike	roll	103	575	0	17	0	permit
1225	mike	move	1	8	8	18	5	permit
1226	mike	move	121	585	0	18	0	permit
1227	mike	move	110	581	0	18	0	permit
1228	mike	roll	125	594	0	19	0	permit
1229	mike	roll	125	595	0	19	0	permit

It was observed that, only the first record of authorization id number (18) is associated with a proper move of five places after rolling the dice. However, the rest of records indicate a misuse actions by moving rolling or moving with a zero roll_id and a zero roll value. One strange observation is the action associated with these records, where sometimes the action is "roll" with zero roll values !!

Then I tried to check the consistency between roll values in three main places:

1. **Roll table**
2. **Move table**
3. **Authorization table**

First for authz_id = 17:

1. Roll table represents that authz_id 17 is associated with game 1 and roll id 8.

roll_id	game_id	value	authz_id	last_updated
8	1	5	17	2014-05-22 14:25:50

2. However, move table represents that authz_id 17 is associated with game 103 and move_id 573 and 575 with move to the game last position without rolling the dice. Whereas, the above information from roll table are associated actually with authz_id 18.

move_id	game_id	roll_id	turn	start_pos	end_pos	authz_id	last_updated
573	103	0	2	4	64	17	2014-06-13 12:27:57
575	103	0	4	8	64	17	2014-06-13 12:30:25

Search: roll_id = 8

move_id	game_id	roll_id	turn	start_pos	end_pos	authz_id	last_updated
8	1	8	8	54	59	18	2014-05-22 14:25:51

3. Authorisation table represents that authz_id 17 is linked to a roll action which is consistent with the information from roll, but not with the move table that reflects a move action!

authz_id	user_id	identity_id	target	action	decision	time_requested	time_responded	last_updated
17	mike	cn=mike,ou=users,o=saaf,c=gb	http://game	roll	permit	1400765149	1400765149	2014-05-22 14:25:49

Second, for authz_id = 18:

1. Roll table represents that authz_id 18 is not associated any roll values

Search: authz_id = 18

roll_id	game_id	value	authz_id	last_updated

4. However, move table represents that authz_id 18 is associated with game 1, 110 and 121 and move_id 8,581 and 585 with one legitimate move by 5 places as per roll value. The other two records indicates that, the player moved illegally without rolling the dice.

Search: authz_id = 18

move_id	game_id	roll_id	turn	start_pos	end_pos	authz_id	last_updated
8	1	8	8	54	59	18	2014-05-22 14:25:51
581	110	0	1	1	64	18	2014-06-13 12:49:18
585	121	0	0	1	6	18	2014-06-13 13:20:31

5. Authorisation table represents that authz_id 18 is linked to a move action which is consistent with the first record of move table

Search: authz_id = 18

authz_id	user_id	identity_id	target	action	decision	time_requested	time_responded	last_updated
18	mike	cn=mike,ou=users,o=saaf,c=gb	http://game	move	permit	1400765151	1400765151	2014-05-22 14:25:51

Third, for authz_id = 19:

1. Roll table represents that authz_id 19 is associated with game 1 and roll_id 9.

Search: authz_id = 19

roll_id	game_id	value	authz_id	last_updated
9	1	1	19	2014-05-22 14:25:52

2. However, move table represents that authz_id 18 is associated with different game (125) and move_id 594, 595 with move to random positions without rolling the dice.

Search: authz_id = 19

move_id	game_id	roll_id	turn	start_pos	end_pos	authz_id	last_updated
594	125	0	1	1	62	19	2014-06-13 13:27:42
595	125	0	2	62	64	19	2014-06-13 13:28:32

3. Authorisation table represents that authz_id 19 is linked to a roll action which is consistent with the information from roll, but not with the move table that reflects a move action!

Search: authz_id = 19

authz_id	user_id	identity_id	target	action	decision	time_requested	time_responded	last_updated
19	mike	cn=mike,ou=users,o=saaf,c=gb	http://game	roll	permit	1400765151	1400765151	2014-05-22 14:25:51

- **In summary, there some re-used authorization ids (requests) that are actually requested in one gamed and reused maliciously in different games. This may lead to**

false values in the combined table of both game and authorization features.

In this Case we have three solutions for this kind of attack:

1. One solution for this is to keep the legitimate values where the data in both move and authorization tables are consistent and remove the illegitimate values that cause inconsistency in records between move and authorization tables. I think we will end up by only keeping two consistent and legitimate records (first record of auth_id =18 and first record of authz_id=2098).
2. OR simply all the data including the duplicated authz_id records in order to avoid losing the data and be able to analyse this type of attack.
3. If the solutions caused any inconsistency in the data, delete the full records related of the duplicated authz_id including the legitimate ones to avoid biasing the data.

Experiment No: 14

Objective:

- *Redo – Experiment 12, but keep a distinct authz_id*

Combine both authorization log and game application log features into one dataset file and classify them with Naïve Bayes algorithm as one dataset, which are:

1. **Deny Count : that represents if the user exceeds a threshold limit of Denies (5), it will be classified as misuse**
2. **Count Number of Consecutive Actions: if any action is repeated consecutively and exceeds a certain limit (10) , it will be classified as misuse**
3. **Count number of Unknown Actions: if the number of unknown actions exceeds a certain limit (3), it will be classified as misuse**
4. **RD: Consecutive number of rolling the dice before actual moves.**
5. **MP: Number of places of moving the piece that exceeds the dice rolls value.**
6. **The accumulative value of the number illegal places the user moved to, or failed to move to.**
7. **The accumulative value of the number of consecutive number of rolls before the player moves.**

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **SAAF Database: Game Logs**
- **Microsoft Excel Spread Sheet.**
- **Weka Tool.**

Procedure:

- **Using Sequel Pro, create a new table to store the combine features information and call it as AllFeatures.**

```
CREATE TABLE AllFeatures
(
        id int not NULL AUTO_INCREMENT,
        user_id varchar(300),
        action varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
ILLEGALMOVE_Value int,
rollsBeforeMove int,
MP_misuse int,
RD_misuse int,
misuse_count_per_move  int,
CUM_MP int,
CUM_RD int,
PRIMARY KEY (id)
)
```

- **Write an SQL query statement that select and combine the required information for both application lavel features and authorisation features based on the common authz_id.**
- **GROUP the values by enforcing only DISTINCT authz_id records to be displayed**
- **Insert the selected values in to the created AllFeatures and associate them with their relevant information.**

```
INSERT
INTO AllFeatures (user_id,action, game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)

SELECT authorisations.user_id,authorisations.action, featuresTable.game_id,
featuresTable.move_id,featuresTable.roll_id,
authorisations.authz_id, featuresTable.roll_value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
```

```
FROM `authorisations`
LEFT OUTER JOIN featuresTable
ON featuresTable.`authz_id`=authorisations.`authz_id`
-- GROUP BY enforcing only DISTINCT authz_id records to be displayed
GROUP BY authorisations.authz_id
ORDER BY authorisations.user_id, authorisations.authz_id
```

- Then, export the ALLFeatures table into an excel file to perform the features calculations.
- Prior injecting the extracted seven features into Weka machine learning tool, train the data composed of the seven features where tool should detects an abuse in the following conditions:
  - *If Deny Count is greater than or equal to 5, OR*
  - *If Count Number of Consecutive Actions is greater than or equal to 10, OR*
  - *If the Countof Uknown Actions is greater than or equal to 3, OR*
  - *If the user does not land on the right position with illegal move value not equal to 0, OR*
  - *If the user rolls the dice consecutively more than one, OR*
  - *If the accumulative value of MP is greater than 21, OR*
  - *If the accumulative value of RD is greater than or equals to 10.*
- Export the resulting table into a CSV file (step 3 features only) , and inject the data file into weka tool.
- Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.
- Evaluate the selected three attributes using InfoGain and Ranker method, to evaluate the worth of each attribute in the classification process.

Results & Discussion of Results:

- By having a distinct authorization id, the total number of records obtained is 2430, which corresponds the total number of records in the authorisation table.
- Classifier output:
- === Evaluation on test split ===

  === Summary ===
  - Correctly Classified Instances 668 91.6324 %
  - Incorrectly Classified Instances 61 8.3676 %
  - Kappa statistic 0.7513
  - Mean absolute error 0.0833

- Root mean squared error 0.2655
- Relative absolute error 23.4384 %
- Root relative squared error 62.7823 %
- Total Number of Instances 729

- === Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.971	0.265	0.923	0.971	0.947	0.964	NO
0.735	0.029	0.887	0.735	0.804	0.964	YES
Weighted Avg. 0.916	0.21	0.915	0.916	0.913	0.964	

=== Confusion Matrix ===

a	b	<-- classified as
543	16	a = NO
45	125	b = YES

The results reveals that there are (Out of 729 test data) :

- 543 true positives (actual non-abuse that were correctly classified as NO)
- 45 false positives (abuse that were incorrectly marked as Non-abuse)
- 16 false negatives (Non-abuse that were incorrectly marked as abuse)
- 125 true negatives (all the remaining instances, correctly classified abuse YES)
- High recall indicates that there are less false negatives.
- With reference to classifier model output in the below figure, the classifier model output indicates the precision of all authorisation three feature is equals to 1 for both classes (YES and NO), whereas application features have a higher precision rates starting from 7.3333 for ILLEGALMOVE_Value followed by 2.7727 for its accumulative value and 2.0476 for RD accumulative value. Then, ending with 1.1667 for rollsBeforeMove feature.
- According the attributes evaluation, the features are ranked according their importance in the classification as follows:

1. Count Number of Consecutive Actions (0. 3715)
2. RD accumulative value (0.1993)
3. Number of Denies exceeding the threshold (0.1461)
4. MP accumulative value (0.1374)
5. Number of rolls Before Move (0.0768)

6. **ILLEGALMOVE_Value for the piece (0.0735)**
7. **Count number of Uknown Actions (0.0707)**

Which reflects the high importance of the Number of Consecutive Actions counter in the classification followed by RD accumulative value.

Figure 1: Classifier model output

Classifier output

```
ILLEGALMOVE_Value
  mean                                    0 -0.0394
  std. dev.                          1.2222  4.6108
  weight sum                           1871     559
  precision                          7.3333  7.3333

rollsBeforeMove
  mean                               0.3467  0.7346
  std. dev.                          0.5332  1.5638
  weight sum                           1871     559
  precision                          1.1667  1.1667

CUM_MP
  mean                               1.4938  6.0712
  std. dev.                          3.0682 10.2813
  weight sum                           1871     559
  precision                          2.7727  2.7727

CUM_RD
  mean                               1.2268   6.326
  std. dev.                          2.0346  7.9938
  weight sum                           1871     559
  precision                          2.0476  2.0476

Deny Count
  mean                               0.2646  3.6512
  std. dev.                          0.7548  6.9128
  weight sum                           1871     559
  precision                             1       1

Count Number of Consecutive  Actions
  mean                               1.9097 12.0215
  std. dev.                          2.4236  10.373
  weight sum                           1871     559
  precision                             1       1

Count Uknown Actions
  mean                               0.0283  2.1163
  std. dev.                          0.1842  5.6606
  weight sum                           1871     559
  precision                             1       1
```

Figure 2: attributes evaluations:

Attribute selection output

```
Evaluator:    weka.attributeSelection.InfoGainAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     step 3 features only
Instances:    2430
Attributes:   8
              ILLEGALMOVE_Value
              rollsBeforeMove
              CUM_MP
              CUM_RD
              Deny Count
              Count Number of Consecutive  Actions
              Count Uknown Actions
              Abuse
Evaluation mode:evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 8 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.3715   6 Count Number of Consecutive  Actions
 0.1993   4 CUM_RD
 0.1461   5 Deny Count
 0.1374   3 CUM_MP
 0.0768   2 rollsBeforeMove
 0.0735   1 ILLEGALMOVE_Value
 0.0707   7 Count Uknown Actions

Selected attributes: 6,4,5,3,2,1,7 : 7
```

Experiment No: 15

Objective:

- *Redo – Experiment 13 without removing the reused authorisation ids, in order to keep track of this kind of attack.*
- **Combine both authorization log and game application log features into one dataset file and classify th/em with Naïve Bayes algorithm as one dataset, which are:**
  1. **Deny Count : that represents if the user exceeds a threshold limit of Denies (5), it will be classified as misuse**
  2. **Count Number of Consecutive Actions: if any action is repeated consecutively and exceeds a certain limit (10), it will be classified as misuse**
  3. **Count number of Unknown Actions: if the number of unknown actions exceeds a certain limit (3), it will be classified as misuse**
  4. **RD: Consecutive number of rolling the dice before actual moves.**
  5. **MP: Number of places of moving the piece that exceeds the dice rolls value.**
  6. **The accumulative value of the number illegal places the user moved to, or failed to move to.**
  7. **The accumulative value of the number of consecutive number of rolls before the player moves.**

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **SAAF Database: Game Logs**
- **Microsoft Excel Spread Sheet.**
- **Weka Tool.**

Procedure:

- **In order to apply a Full join for all information associated with game data with authorization data, the following SQL query was used to extract desired information and insert the results in a new table :**

```
-- create a new table called combinedTable as a test for step 3
CREATE TABLE combined_step3
(
    id int not NULL AUTO_INCREMENT,
    user_id varchar(300),
```

```
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
            ILLEGALMOVE_Value int,
            rollsBeforeMove int,
           MP_misuse int, -- to be set if ILLEGALMOVE_Value not zero
           RD_misuse int,  -- to be set if  rollsBeforeMove not equals to one
         misuse_count_per_move  int,
          CUM_MP int,
         CUM_RD int,
         PRIMARY KEY (id)
    )

INSERT
INTO combined_step3 (user_id, action, game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)

SELECT authorisations.user_id,authorisations.action, roll.game_id, featuresTable.move_id,roll.roll_id,
authorisations.authz_id, roll.value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
LEFT OUTER JOIN featuresTable ON featuresTable.`authz_id`=authorisations.`authz_id`
LEFT OUTER JOIN roll ON roll.`authz_id`=authorisations.`authz_id`
ORDER BY authorisations.user_id, authorisations.authz_id
```

Results & Discussion of Results:

The total number of authorization table is 2430, however the combined table reveals the existence of 2526 records which reveals that roll table contains some reused (duplicated) authorization ids beside the 13 duplicated records observed in the move table. They can be viewed using below SQL query as showed in the figure:

```
SELECT
    authz_id,  game_id, roll_id, COUNT(*)
FROM
    roll
GROUP BY
    authz_id
HAVING
    COUNT(*) > 1
```

Query Favorites ▾ Query History ▾ Run Current

authz_id	game_id	roll_id	COUNT(*)
742	40	306	84

Above figure reveals that, there 84 records of the same authorisation id (742). Which can be added up with move table reused authz_id to form the right number of records as obtained from the first query.

One problem encountered while checking the information provided by the combined table. There is some missing information associated with the roll actions, where they have to be obtained from the roll table to have an idea about the illegal rolls that are not associated with any move recorded in move table.

In order to resolve this issue, a new joined table was created that combines the move and roll actions related information, so that they contain all rolls that does not correspond with move action as well and can be recalled in the joining phase.

```
CREATE TABLE move_roll (
  `roll_id` int(11) DEFAULT NULL,
  authz_id int(11) DEFAULT NULL,
  `game_id` int(11) DEFAULT NULL,
  `roll_value` int(11) DEFAULT NULL,
    `move_id` int(11) DEFAULT NULL,
  `start_pos` int(11) DEFAULT NULL,
  `end_pos` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT
INTO move_roll (roll_id,authz_id, game_id, roll_value, move_id, start_pos, end_pos)

SELECT roll.roll_id, roll.authz_id, roll.game_id, roll.value, move.move_id, move.start_pos, move.end_pos
```

FROM roll

left outer JOIN move ON move.roll_id = roll.roll_id

Test joining move_roll table with the already extracted features from the game in a new table called Game_Features:

```
-- create a new table called Game_Features
CREATE TABLE Game_Features
(
        id int not NULL AUTO_INCREMENT,
        user_id varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
       ILLEGALMOVE_Value int,
       rollsBeforeMove int,
      MP_misuse int, -- to be set if ILLEGALMOVE_Value not zero
      RD_misuse int, , -- to be set if  rollsBeforeMove not equals to one
    misuse_count_per_move  int,
     CUM_MP int,
    CUM_RD int,
    PRIMARY KEY (id)
 )
```

INSERT

INTO Game_Features (user_id, game_id, move_id,roll_id, authz_id, roll_value, decision, ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)

SELECT authorisations.user_id, move_roll.game_id, move_roll.move_id, move_roll.roll_id, move_roll.authz_id, move_roll.roll_value, authorisations.decision, MP5.ILLEGALMOVE_Value, RD.rollsBeforeMove, if(MP5.ILLEGALMOVE_Value!=0, 1, 0) as MP_misuse, if(RD.rollsBeforeMove!=1, 1, 0) as RD_misuse, 0 as misuse_count_per_move

FROM move_roll

LEFT OUTER JOIN authorisations ON authorisations.`authz_id`= move_roll.`authz_id`

```
LEFT OUTER JOIN MP5 ON MP5.move_id = move_roll.move_id
LEFT OUTER JOIN RD ON RD.roll_id = move_roll.roll_id

ORDER BY authorisations.user_id, authorisations.authz_id
```

- **SQL JOINING COMMAND FOR BOTH GAME AND AUTHORIZATION INFORMATIONS:**

```
SELECT authorisations.user_id,authorisations.action, game.game_id, move.move_id,roll.roll_id,
authorisations.authz_id, roll.value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
LEFT OUTER JOIN featuresTable ON featuresTable.`authz_id`=authorisations.`authz_id`
LEFT OUTER JOIN roll ON roll.`authz_id`=authorisations.`authz_id`
LEFT OUTER JOIN game ON game.`authz_id`=authorisations.`authz_id`
LEFT OUTER JOIN move ON move.`authz_id`=authorisations.`authz_id`

UNION
SELECT authorisations.user_id,authorisations.action, game.game_id, move.move_id,roll.roll_id,
authorisations.authz_id, roll.value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
RIGHT OUTER JOIN featuresTable ON featuresTable.`authz_id`=authorisations.`authz_id`
RIGHT OUTER JOIN roll ON roll.`authz_id`=authorisations.`authz_id`
RIGHT OUTER JOIN game ON game.`authz_id`=authorisations.`authz_id`
RIGHT OUTER JOIN move ON move.`authz_id`=authorisations.`authz_id

ORDER BY authorisations.user_id, authorisations.authz_id
```

- **After trying combining the results, it has been observed that, it is difficult to match the details of roll action and move action sometimes as they has different authorization requests. However, the same authorization request is matched with both move_id and roll_id, which makes the matching function is difficult in this case.**

  >> This point as been checked by displaying the roll_id from both move and roll table at the same time, and the difference can be noticed in the below output:

```
SELECT authorisations.user_id,authorisations.action, roll.game_id,featuresTable.game_id,
```

```
featuresTable.move_id, roll.roll_id as Roll_table_roll_id,featuresTable.roll_id as Move_roll_id,
authorisations.authz_id, roll.value, featuresTable.roll_value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove, featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
LEFT OUTER JOIN featuresTable ON featuresTable.`authz_id`=authorisations.`authz_id`
LEFT OUTER JOIN roll ON roll.`authz_id`=authorisations.`authz_id`

ORDER BY authorisations.user_id, authorisations.authz_id
```

user_id	action	game_id	move_id	Roll_table_roll_id	Move_roll_id	authz_id	value	decision	ILLEGALMOVE_Value	rollsBeforeMove	MP_misuse	RD_misuse	n
	bonus	NULL	NULL	NULL	NULL	1848	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1849	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1850	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1851	NULL	deny	NULL	NULL	NULL	NULL	
	ro	NULL	NULL	NULL	NULL	1852	NULL	deny	NULL	NULL	NULL	NULL	
	ro	NULL	NULL	NULL	NULL	1853	NULL	deny	NULL	NULL	NULL	NULL	
	ro	NULL	NULL	NULL	NULL	1854	NULL	deny	NULL	NULL	NULL	NULL	
		NULL	NULL	NULL	NULL	1855	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1856	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1857	NULL	deny	NULL	NULL	NULL	NULL	
	move	NULL	NULL	NULL	NULL	1858	NULL	deny	NULL	NULL	NULL	NULL	
	roll	NULL	NULL	NULL	NULL	1859	NULL	deny	NULL	NULL	NULL	NULL	
1234	start	NULL	NULL	NULL	NULL	2033	NULL	permit	NULL	NULL	NULL	NULL	
1234	roll	NULL	NULL	1019	NULL	2034	4	permit	NULL	NULL	NULL	NULL	
1234	move	138	629	NULL	1019	2035	NULL	permit	0	1	0	0	
1234	roll	NULL	NULL	1020	NULL	2036	2	permit	NULL	NULL	NULL	NULL	
1234	bonus	NULL	NULL	NULL	NULL	2037	NULL	permit	NULL	NULL	NULL	NULL	
1234	move	138	630	NULL	1020	2038	NULL	permit	0	1	0	0	
1234	roll	NULL	NULL	1021	NULL	2039	5	permit	NULL	NULL	NULL	NULL	
1234	ladder	NULL	NULL	NULL	NULL	2040	NULL	permit	NULL	NULL	NULL	NULL	
1234	move	138	631	NULL	1021	2041	NULL	permit	0	1	0	0	
1234	roll	NULL	NULL	1022	NULL	2042	2	permit	NULL	NULL	NULL	NULL	
1234	move	138	632	NULL	1022	2043	NULL	permit	0	1	0	0	

Hence, it has been decided to just focus on the move level features and ignores the display for associated roll information for time being because of time constraint.

Experiment No: 16

Objective:

- *Classify the full combined features using Naïve Bayes algorithm as one dataset.*
- *Analyse the classifier results and compare them with the last independent classifier results.*

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **SAAF Database: Game Logs**
- **Microsoft Excel Spread Sheet.**
- **Weka Tool.**

Procedure:

- **It has been decided to keep all records (including the reused athuz_id related records) in a combined table using below SQL Query:**

```
INSERT
INTO combinedTable (user_id, action, game_id, move_id,roll_id, authz_id, roll_value,
decision, ILLEGALMOVE_Value, rollsBeforeMove, MP_misuse, RD_misuse,
misuse_count_per_move)
SELECT authorisations.user_id,authorisations.action, featuresTable.game_id,
featuresTable.move_id,featuresTable.roll_id,
authorisations.authz_id, featuresTable.roll_value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove,
featuresTable.MP_misuse,
featuresTable.RD_misuse, featuresTable.misuse_count_per_move
FROM `authorisations`
LEFT OUTER JOIN featuresTable
ON featuresTable.`authz_id`=authorisations.`authz_id`
ORDER BY authorisations.user_id, authorisations.authz_id, featuresTable.move_id
```

- **Preprocess the combinedTable that contains 2443 records obtained after performing the outer join between game features and authorization table features.**
- **Prior injecting the extracted combined features in to Weka machine learning tool, train the data composed of the seven features such that tool should detects abuse if the following scenarios becomes true:**

1. **If Deny Count is greater than or equal to 5, OR**
2. **If Count Number of Consecutive Actions is greater than or equal to 10, OR**
3. **If the Countof Uknown Actions is greater than or equal to 3, OR**
4. **If the user does not land on the right position with illegal move value not equal to 0, OR**
5. **If the user rolls the dice consecutively more than one, OR**
6. **If the accumulative value of MP is greater than 21, OR**
7. **If the accumulative value of RD is greater than or equals to 10.**

- **Export the resulting table into a CSV file (step 3 full combine) , and inject the data file into weka tool.**
- **Run Naïve Bayes Classifier with a percentage split of 70 % as a training dataset and 30% as test data set.**
- **Evaluate the selected three attributes using InfoGain and Ranker method, to evaluate the worth of each attribute in the classification process.**

Results & Discussion of Results:

- Out of 4443 data set, 13 instances represented a reused authorization ids extracted from the game data, which reflects that a user attempts to do illegal action using an existing authorization request admitted earlier to another user or a game.
- Classifier output:
- === Evaluation on test split ===

  === Summary ===
  - Correctly Classified Instances 668 91.6324 %
  - Incorrectly Classified Instances 61 8.3676 %
  - Kappa statistic 0.7513
  - Mean absolute error 0.0833
  - Root mean squared error 0.2655
  - Relative absolute error 23.4384 %

- Root relative squared error 62.7823 %
- Total Number of Instances 729

- === Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.971	0.265	0.923	0.971	0.947	0.964	NO
0.735	0.029	0.887	0.735	0.804	0.964	YES
Weighted Avg. 0.916	0.21	0.915	0.916	0.913	0.964	

=== Confusion Matrix ===

a	b	<-- classified as
543	16	a = NO
45	125	b = YES

The results reveals that there are (Out of 729 test data) :

- 543 true positives (actual non-abuse that were correctly classified as NO)
- 45 false positives (abuse that were incorrectly marked as Non-abuse)
- 16 false negatives (Non-abuse that were incorrectly marked as abuse)
- 125 true negatives (all the remaining instances, correctly classified abuse YES)
- High recall indicates that there are less false negatives.

- With reference to classifier model output in the below figure, the classifier model output indicates the precision of all authorisation three feature is equals to 1 for both classes (YES and NO), whereas application features have a higher precision rates starting from 7.3333 for ILLEGALMOVE_Value followed by 2.7727 for its accumulative value and 2.0476 for RD accumulative value. Then, ending with 1.1667 for rollsBeforeMove feature.

- According the attributes evaluation, the features are ranked according their importance in the classification as follows:

8. Count Number of Consecutive Actions (0. 3715)
9. RD accumulative value (0.1993)
10. Number of Denies exceeding the threshold (0.1461)
11. MP accumulative value (0.1374)

12. Number of rolls Before Move (0.0768)

13. ILLEGALMOVE_Value for the piece (0.0735)

14. Count number of Uknown Actions (0.0707)

Which reflects the high importance of the Number of Consecutive Actions counter in the classification followed by RD accumulative value.

Figure 1: Classifier model output

Classifier output

ILLEGALMOVE_Value		
mean	0	-0.0394
std. dev.	1.2222	4.6108
weight sum	1871	559
precision	7.3333	7.3333
rollsBeforeMove		
mean	0.3467	0.7346
std. dev.	0.5332	1.5638
weight sum	1871	559
precision	1.1667	1.1667
CUM_MP		
mean	1.4938	6.0712
std. dev.	3.0682	10.2813
weight sum	1871	559
precision	2.7727	2.7727
CUM_RD		
mean	1.2268	6.326
std. dev.	2.0346	7.9938
weight sum	1871	559
precision	2.0476	2.0476
Deny Count		
mean	0.2646	3.6512
std. dev.	0.7548	6.9128
weight sum	1871	559
precision	1	1
Count Number of Consecutive Actions		
mean	1.9097	12.0215
std. dev.	2.4236	10.373
weight sum	1871	559
precision	1	1
Count Uknown Actions		
mean	0.0283	2.1163
std. dev.	0.1842	5.6606
weight sum	1871	559
precision	1	1

Figure 2: attributes evaluations:

Attribute selection output

```
Evaluator:    weka.attributeSelection.InfoGainAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     step 3 features only
Instances:    2430
Attributes:   8
              ILLEGALMOVE_Value
              rollsBeforeMove
              CUM_MP
              CUM_RD
              Deny Count
              Count Number of Consecutive  Actions
              Count Uknown Actions
              Abuse
Evaluation mode:evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 8 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.3715   6 Count Number of Consecutive  Actions
 0.1993   4 CUM_RD
 0.1461   5 Deny Count
 0.1374   3 CUM_MP
 0.0768   2 rollsBeforeMove
 0.0735   1 ILLEGALMOVE_Value
 0.0707   7 Count Uknown Actions

Selected attributes: 6,4,5,3,2,1,7 : 7
```

Experiment No: 17

Objective:

- *Redo – Experiment 12, witha distinct authz_id and eliminate the illegitimate records with reused authorization ids.*
  - Combine both authorization log and game application log features into one dataset file and classify them with Naïve Bayes algorithm as one dataset, which are:
    1. Deny Count : that represents if the user exceeds a threshold limit of Denies (5), it will be classified as misuse
    2. Count Number of Consecutive Actions: if any action is repeated consecutively and exceeds a certain limit (10) , it will be classified as misuse
    3. Count number of Unknown Actions: if the number of unknown actions exceeds a certain limit (3), it will be classified as misuse
    4. RD: Consecutive number of rolling the dice before actual moves.
    5. MP: Number of places of moving the piece that exceeds the dice rolls value.
    6. The accumulative value of the number illegal places the user moved to, or failed to move to.
    7. The accumulative value of the number of consecutive number of rolls before the player moves.

Tools and Data used:

- Sequel Pro Software (Structured Query Language)
- SAAF Database: Game Logs
- Microsoft Excel Spread Sheet.
- Weka Tool.

Procedure:

As discussed in Experiment 12

Results & Discussion of Results:

- By having a distinct authorization id and removing illegitimate authorization ids records, the total number of records obtained is 2428, which corresponds the total number of records in the authorisation table.

  === Evaluation on test split ===

  === Summary ===

Correctly Classified Instances	683	93.8187 %
Incorrectly Classified Instances	45	6.1813 %
Kappa statistic	0.8156	
Mean absolute error	0.0656	
Root mean squared error	0.2333	
Relative absolute error	18.4686 %	
Root relative squared error	55.2678 %	
Total Number of Instances	728	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.986	0.219	0.937	0.986	0.961	0.973	NO
	0.781	0.014	0.943	0.781	0.854	0.973	YES
Weighted Avg.	0.938	0.171	0.938	0.938	0.936	0.973	

=== Confusion Matrix ===

```
   a   b   <-- classified as
 551   8 |   a = NO
  37 132 |   b = YES
```

The results reveals that there are (Out of 728 test data) :

- 551 true positives (actual non-abuse that were correctly classified as NO)

- **37 false positives (abuse that were incorrectly marked as Non-abuse)**
- **8 false negatives (Non-abuse that were incorrectly marked as abuse)**
- **132 true negatives (all the remaining instances, correctly classified abuse YES)**
- **High recall indicates that there are less false negatives.**
- **With reference to classifier model output in the below figure, the classifier model output indicates the precision of all authorisation three feature is equals to 1 for both classes (YES and NO), whereas application features have a higher precision rates starting from 7.3333 for ILLEGALMOVE_Value followed by 2.7727 for its accumulative value and 2.0476 for RD accumulative value. Then, ending with 1.1667 for rollsBeforeMove feature.**

- **According the attributes evaluation, the features are ranked according their importance in the classification as follows:**

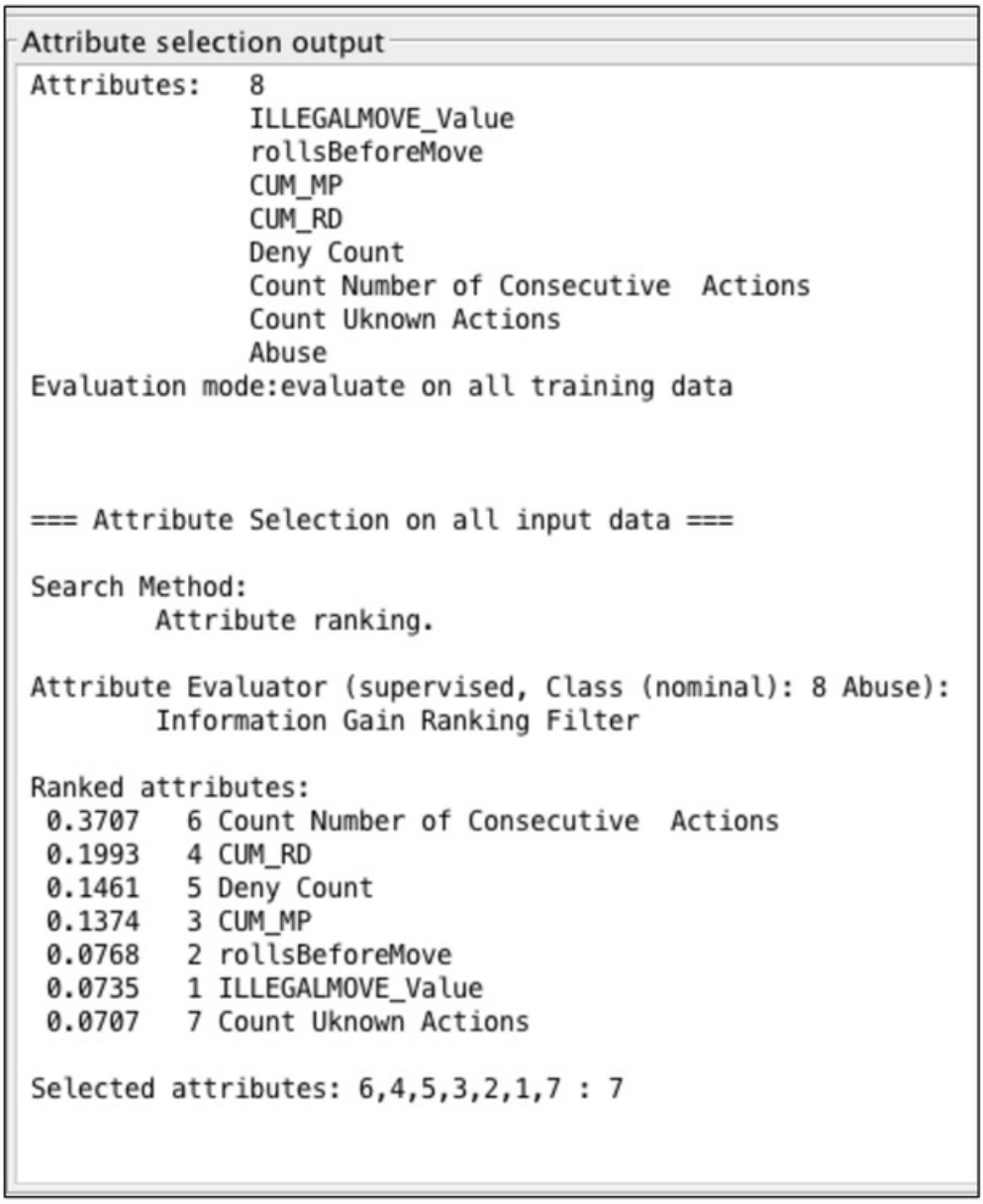
15. **Count Number of Consecutive Actions (0. 3707)**
16. **RD accumulative value (0.1993)**
17. **Number of Denies exceeding the threshold (0.1461)**
18. **MP accumulative value (0.1374)**
19. **Number of rolls Before Move (0.0768)**
20. **ILLEGALMOVE_Value for the piece (0.0735)**
21. **Count number of Uknown Actions (0.0707)**

Which reflects the high importance of the Number of Consecutive Actions counter in the classification followed by RD accumulative value.

Figure 1: Classifier model output

```
Naive Bayes Classifier

                                          Class
Attribute                                   NO     YES
                                         (0.77)  (0.23)
=============================================================
ILLEGALMOVE_Value
  mean                                       0 -0.0394
  std. dev.                             1.2222  4.6108
  weight sum                              1869     559
  precision                             7.3333  7.3333

rollsBeforeMove
  mean                                  0.3458  0.7346
  std. dev.                             0.5328  1.5638
  weight sum                              1869     559
  precision                             1.1667  1.1667

CUM_MP
  mean                                  1.4954  6.0712
  std. dev.                             3.0694 10.2813
  weight sum                              1869     559
  precision                             2.7727  2.7727

CUM_RD
  mean                                  1.2281   6.326
  std. dev.                             2.0353  7.9938
  weight sum                              1869     559
  precision                             2.0476  2.0476

Deny Count
  mean                                  0.2648  3.6512
  std. dev.                             0.7552  6.9128
  weight sum                              1869     559
  precision                                  1       1

Count Number of Consecutive  Actions
  mean                                  1.9829 12.0322
  std. dev.                              2.396 10.3616
  weight sum                              1869     559
  precision                                  1       1

Count Uknown Actions
  mean                                  0.0284  2.1163
  std. dev.                             0.1843  5.6606
  weight sum                              1869     559
  precision                                  1       1
```

Figure 2: Classifier attributes evaluator



Experiment No: 18

Objective:

- *Use Live data to be classified by Naïve Bayes Classifier for already extracted game feature, where classifier has been trained already from the data gathered in the last experiments.*

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **SAAF Database: Game Logs**
- **Microsoft Excel Spread Sheet.**
- **Weka Tool.**

Procedure:

- **A newly data has been provided for new games played, so that they can be used in classifying the data with out providing an actual prediction of whether those records are abuse or not.**
- **Delete already trained and tested records.**

```
DELETE FROM `move`
WHERE move_id <= 758

DELETE FROM `roll`
WHERE roll_id <= 1219

DELETE FROM `game`
WHERE game_id <= 169
```

So that the number of new instances became:
335 new moves
123 new games played
277 new rolls

- **Hence, the experiments have been repeated to acquire the values associated with same extracted game and authorization features besides the combined table.**

```
INSERT
INTO featuresTable (user_id,game_id, move_id,roll_id, authz_id, roll_value, decision, ILLEGALMOVE_Value,
rollsBeforeMove, MP_misuse, RD_misuse, misuse_count_per_move)


SELECT  authorisations.user_id, MP.game_id, MP.move_id, MP.roll_id, MP.authz_id, MP.roll_value,
authorisations.decision, MP.ILLEGALMOVE_Value, RD.rollsBeforeMove, if(MP.ILLEGALMOVE_Value!=0, 1, 0 ) as
MP_misuse, if(RD.rollsBeforeMove!=1, 1, 0) as RD_misuse, 0 as misuse_count_per_move
```

```
FROM MP, RD,  authorisations
WHERE MP.move_id = RD.move_id
AND authorisations.authz_id =  MP.authz_id
ORDER BY authorisations.user_id, authorisations.authz_id
```

- **334 records obtained from the game features table where it has been observed that, one record is missing in the game features table because its associated authz_id= 0, which might give indication about one possible attack.**
- **910 records have been obtained from the authorization features table.**

Discussion of results

*** As classifier understand from previous data, it was able to judge if the behavior is abuse or not based on anomalies observed.**

Example on a subset of live tested data:

=== Predictions on test split ===

inst#, actual, predicted, error, probability distribution

(ILLEGALMOVE_Value,rollsBeforeMove,CUM_MP,CUM_RD,Deny Count,Count Number of Consecutive Actions,Count Uknown Actions)

102	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
103	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
104	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
105	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
106	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
107	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
108	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
109	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
110	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
111	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
112	?	1:NO	+	*0.959	0.041 (0,1,7,0,3,0,0)
113	?	1:NO	+	*0.98	0.02 (0,0,7,0,3,0,0)
114	?	2:YES	+	0.226	*0.774 (0,0,7,0,4,0,0)
115	?	2:YES	+	0	*1 (0,0,7,0,5,1,0)
116	?	2:YES	+	0	*1 (0,0,7,0,6,2,0)
117	?	2:YES	+	0	*1 (0,0,7,0,6,2,0)
118	?	2:YES	+	0	*1 (0,2,7,1,6,2,0)
119	?	2:YES	+	0	*1 (0,0,7,0,7,2,0)
120	?	2:YES	+	0	*1 (0,0,7,0,8,3,0)
121	?	2:YES	+	0	*1 (0,0,7,0,9,4,0)
122	?	2:YES	+	0	*1 (0,0,7,0,10,5,0)
123	?	2:YES	+	0	*1 (0,0,7,0,11,6,0)
124	?	1:NO	+	*1	0 (0,0,0,0,0,0,0)

Figure 1: Classifier model output

Classifier output

```
ILLEGALMOVE_Value
  mean                                    0 -0.0394
  std. dev.                          1.2222  4.6108
  weight sum                           1871     559
  precision                          7.3333  7.3333

rollsBeforeMove
  mean                               0.3467  0.7346
  std. dev.                          0.5332  1.5638
  weight sum                           1871     559
  precision                          1.1667  1.1667

CUM_MP
  mean                               1.4938  6.0712
  std. dev.                          3.0682 10.2813
  weight sum                           1871     559
  precision                          2.7727  2.7727

CUM_RD
  mean                               1.2268   6.326
  std. dev.                          2.0346  7.9938
  weight sum                           1871     559
  precision                          2.0476  2.0476

Deny Count
  mean                               0.2646  3.6512
  std. dev.                          0.7548  6.9128
  weight sum                           1871     559
  precision                             1       1

Count Number of Consecutive  Actions
  mean                               1.9097 12.0215
  std. dev.                          2.4236  10.373
  weight sum                           1871     559
  precision                             1       1

Count Uknown Actions
  mean                               0.0283  2.1163
  std. dev.                          0.1842  5.6606
  weight sum                           1871     559
  precision                             1       1
```

Figure 2: attributes evaluations:

Attribute selection output

```
Evaluator:    weka.attributeSelection.InfoGainAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     step 3 features only
Instances:    2430
Attributes:   8
              ILLEGALMOVE_Value
              rollsBeforeMove
              CUM_MP
              CUM_RD
              Deny Count
              Count Number of Consecutive  Actions
              Count Uknown Actions
              Abuse
Evaluation mode:evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 8 Abuse):
        Information Gain Ranking Filter

Ranked attributes:
 0.3715   6 Count Number of Consecutive  Actions
 0.1993   4 CUM_RD
 0.1461   5 Deny Count
 0.1374   3 CUM_MP
 0.0768   2 rollsBeforeMove
 0.0735   1 ILLEGALMOVE_Value
 0.0707   7 Count Uknown Actions

Selected attributes: 6,4,5,3,2,1,7 : 7
```

Experiment No: 19

Objective:

- *Fuse the outcome of the first two classifiers from phase 1 and 2*

Tools and Data used:

- **Sequel Pro Software (Structured Query Language)**
- **SAAF Database: Game Logs**
- **Microsoft Excel Spread Sheet.**
- **Weka Tool.**

Procedure:

1) **In order to fuse both classifiers, the common instances between them have be extracted using SQL (which are 757 records)**

```
CREATE TABLE phase4
(
        id int not NULL AUTO_INCREMENT,
        user_id varchar(300),
        action varchar(300),
        game_id int,
        move_id int,
        roll_id int,
        authz_id int,
        roll_value int,
        decision varchar(300),
 ILLEGALMOVE_Value int,
 rollsBeforeMove int,
 CUM_MP int,
 CUM_RD int,
 PRIMARY KEY (id)
 )
INSERT
INTO phase4 (user_id,action, game_id, move_id,roll_id, authz_id, roll_value, decision,
ILLEGALMOVE_Value, rollsBeforeMove)
SELECT authorisations.user_id,authorisations.action, featuresTable.game_id,
featuresTable.move_id,featuresTable.roll_id,
authorisations.authz_id, featuresTable.roll_value, authorisations.decision,
featuresTable.ILLEGALMOVE_Value, featuresTable.rollsBeforeMove
FROM `authorisations`
INNER JOIN featuresTable
ON featuresTable.`authz_id`=authorisations.`authz_id`
ORDER BY authorisations.user_id, authorisations.authz_id
```

Discussion of results

If only common instance between authorization table and game table are classified to predict the abuse on authorization level using three agreed featured from phase 2, the following performance measures were obtained:

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	**227**	**100 %**
Incorrectly Classified Instances	**0**	**0 %**
Kappa statistic	**1**	
Mean absolute error	**0.0795**	
Root mean squared error	**0.133**	
Relative absolute error	**15.91 %**	
Root relative squared error	**26.5977 %**	
Total Number of Instances	**227**	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	NO
	1	0	1	1	1	1	YES
Weighted Avg.	1	0	1	1	1	1	

=== Confusion Matrix ===

```
  a   b   <-- classified as
115   0 |   a = NO
  0 112 |   b = YES
```

Which indicates 100 % accuracy on the results as features attributes are far from the set thresholds.

**** Then by fusing the outcomes of game features and authorization features classifications, and applying simple OR gate on the results. It has been found that out of 227 test instances:**

Game features come with 39 abuses.

Authorization features come with 93 abuses.

Fused OR gate outcome come with 118 abuses originated from both cases.

Which reveals that those two features complement each other and collaborate to extend detection scope to detect more abuses.

Figure 1 : Fusing table

INDEX	*Classifier 1*	*Classifier 2*	*Abuse*
1	NO	YES	YES
2	NO	YES	YES
3	NO	YES	YES
4	NO	YES	YES
5	NO	YES	YES
6	NO	YES	YES
7	NO	YES	YES
8	NO	YES	YES
9	NO	YES	YES
10	NO	YES	YES
11	NO	YES	YES
12	NO	YES	YES
13	NO	YES	YES
14	NO	YES	YES
15	NO	YES	YES
16	NO	YES	YES
17	NO	YES	YES
18	YES	YES	YES
19	YES	YES	YES
20	NO	YES	YES
21	NO	YES	YES
22	NO	NO	NO
23	NO	NO	NO
24	NO	NO	NO
25	NO	NO	NO
26	NO	NO	NO
27	NO	NO	NO
28	NO	NO	NO
29	YES	NO	YES

30	NO	NO	NO
31	NO	NO	NO
32	NO	YES	YES
33	NO	YES	YES
34	YES	YES	YES
35	NO	YES	YES
36	NO	NO	NO
37	NO	NO	NO
38	NO	NO	NO
39	YSE	NO	NO
40	NO	NO	NO
41	NO	NO	NO
42	YES	NO	YES
43	NO	NO	NO
44	YES	NO	YES
45	YES	NO	YES
46	YES	NO	YES
47	NO	NO	NO
48	YES	NO	YES
49	NO	NO	NO
50	YES	NO	YES
51	NO	NO	NO
52	NO	NO	NO
53	NO	YES	YES
54	NO	YES	YES
55	NO	YES	YES
56	YES	YES	YES
57	NO	YES	YES
58	NO	YES	YES
59	YES	YES	YES
60	NO	YES	YES
61	YES	YES	YES
62	NO	YES	YES
63	NO	YES	YES

64	NO	YES	YES
65	NO	YES	YES
66	YES	YES	YES
67	NO	YES	YES
68	NO	YES	YES
69	NO	YES	YES
70	NO	YES	YES
71	YES	YES	YES
72	NO	YES	YES
73	NO	NO	NO
74	NO	NO	NO
75	NO	NO	NO
76	NO	NO	NO
77	NO	NO	NO
78	NO	NO	NO
79	YES	NO	YES
80	YES	NO	YES
81	NO	NO	NO
82	NO	NO	NO
83	NO	NO	NO
84	NO	NO	NO
85	NO	NO	NO
86	NO	NO	NO
87	YES	NO	YES
88	NO	NO	NO
89	YES	NO	YES
90	NO	YES	YES
91	NO	YES	YES
92	NO	YES	YES
93	NO	YES	YES
94	NO	YES	YES
95	NO	YES	YES
96	NO	YES	YES
97	NO	YES	YES

98	NO	YES	YES
99	YES	YES	YES
100	NO	YES	YES
101	NO	YES	YES
102	NO	YES	YES
103	NO	YES	YES
104	NO	YES	YES
105	NO	YES	YES
106	NO	NO	NO
107	NO	NO	NO
108	NO	NO	NO
109	NO	NO	NO
110	NO	NO	NO
111	YES	NO	YES
112	NO	NO	NO
113	NO	NO	NO
114	NO	NO	NO
115	YES	NO	YES
116	NO	NO	NO
117	NO	NO	NO
118	NO	NO	NO
119	YES	NO	YES
120	NO	NO	NO
121	NO	NO	NO
122	YES	NO	YES
123	NO	YES	YES
124	NO	YES	YES
125	YES	YES	YES
126	NO	YES	YES
127	NO	YES	YES
128	YES	YES	YES
129	NO	YES	YES
130	NO	YES	YES
131	NO	YES	YES

132	NO	YES	YES
133	NO	NO	NO
134	YES	NO	YES
135	NO	NO	NO
136	YES	NO	YES
137	NO	NO	NO
138	YES	NO	YES
139	NO	NO	NO
140	NO	NO	NO
141	NO	NO	NO
142	YES	NO	YES
143	NO	NO	NO
144	NO	NO	NO
145	NO	NO	NO
146	NO	NO	NO
147	YES	NO	YES
148	NO	NO	NO
149	YES	NO	YES
150	NO	NO	NO
151	YES	NO	YES
152	NO	NO	NO
153	NO	NO	NO
154	NO	NO	NO
155	NO	NO	NO
156	NO	NO	NO
157	NO	NO	NO
158	NO	NO	NO
159	NO	YES	YES
160	NO	YES	YES
161	YES	YES	YES
162	NO	YES	YES
163	NO	YES	YES
164	NO	YES	YES
165	NO	YES	YES

166	NO	YES	YES
167	NO	YES	YES
168	NO	YES	YES
169	YES	YES	YES
170	NO	YES	YES
171	NO	YES	YES
172	NO	NO	NO
173	NO	NO	NO
174	NO	NO	NO
175	NO	NO	NO
176	NO	NO	NO
177	NO	NO	NO
178	NO	NO	NO
179	NO	NO	NO
180	NO	NO	NO
181	NO	NO	NO
182	NO	YES	YES
183	NO	YES	YES
184	YES	NO	YES
185	NO	NO	NO
186	NO	NO	NO
187	NO	NO	NO
188	YES	NO	YES
189	NO	NO	NO
190	NO	NO	NO
191	NO	NO	NO
192	NO	NO	NO
193	NO	NO	NO
194	NO	NO	NO
195	NO	NO	NO
196	NO	NO	NO
197	NO	NO	NO
198	NO	NO	NO
199	NO	NO	NO

200	NO	NO	NO
201	NO	NO	NO
202	NO	NO	NO
203	NO	NO	NO
204	YES	NO	YES
205	NO	NO	NO
206	NO	NO	NO
207	NO	NO	NO
208	NO	NO	NO
209	NO	NO	NO
210	NO	NO	NO
211	NO	YES	YES
212	NO	YES	YES
213	NO	NO	NO
214	NO	NO	NO
215	NO	NO	NO
216	NO	NO	NO
217	NO	NO	NO
218	NO	NO	NO
219	NO	NO	NO
220	NO	NO	NO
221	NO	NO	NO
222	NO	NO	NO
223	NO	YES	YES
224	NO	YES	YES
225	NO	YES	YES
226	YES	YES	YES
227	NO	YES	YES

A Machine Learning Approach To Detecting Insider Threat In Access Control System

GRADEMARK REPORT

FINAL GRADE

/100

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96

PAGE 97

PAGE 98

PAGE 99

PAGE 100

PAGE 101

PAGE 102

PAGE 103

PAGE 104

PAGE 105

PAGE 106

PAGE 107

PAGE 108

PAGE 109

PAGE 110

PAGE 111

PAGE 112

PAGE 113

PAGE 114

PAGE 115

PAGE 116

PAGE 117

PAGE 118

PAGE 119

PAGE 120

PAGE 121

PAGE 122

PAGE 123

PAGE 124

PAGE 125

PAGE 126

PAGE 127

PAGE 128

PAGE 129

PAGE 130

PAGE 131

PAGE 132

PAGE 133

PAGE 134

PAGE 135

PAGE 136

PAGE 137

PAGE 138

PAGE 139

PAGE 140

PAGE 141

PAGE 142

PAGE 143

PAGE 144

PAGE 145