Insider Threat detection using Isolation Forest

Joakim Bülow
joakim.bulow@gmail.com

Maja Scherman
maja.scherman@gmail.com

June 21, 2018

Master's thesis work carried out at Elastic Mobile Scandinavia AB and Lund University.

Supervisors: Emma Fitzgerald, emma.fitzgerald@eit.lth.se
Richard Niklasson, richard.niklasson@elasticmobile.se

Examiner: Christian Nyberg, christian.nyberg@eit.lth.se

Abstract

In contrast to the need for companies to get real time information about insider threats, there is a privacy and integrity based limitation of what the individual accepts as acceptable surveillance. This creates a problem since performing online surveillance would pose an infringement on the employees privacy and integrity.

Therefore we present a model using Isolation Forest to solve this problem. We focus on analyzing the non-intrusive features in a real time, event based approach. We process our features using periodic features, which we have statistically proven to be more effective than periodic features used with Isolation Forest.

Our results show that by analyzing employees login and logout times, we can detect 76% of all insider threats while only falsely classify 7% of all normal instances. The recall rate, which shows how complete the results are, is 76%.

Keywords: MSc, Insider Threats, Isolation Forest, Machine Learning, Security

Acknowledgements

We would like to thank our supervisor, Emma Fitzgerald, for good feedback and taking her time to assist us throughout the process. We would also like to thank Christian Nyberg for being engaged in the projects process from day one. We would also like to thank Richard Niklasson and the rest of our colleagues for supporting us so we could combine work with studies.

Contents

Popular Summary

Digitalization has brought us great opportunities for economic growth and there has been a global trend for companies to store more and more of their assets and products in digital form for many years. But digitalization also brought new types of risks and vulnerabilities and to stay secure companies needs to invest in countermeasures. Companies are prone to put great recourses into securing their digital perimeters and exposure to the Internet to prevent cyber-crime and digital theft. What is commonly ignored is the possibility of threats from within the company perimeters, so called insider threats. What if we could detect and prevent insider threats before they ever occurred? For example, if an employee feels let down by the company and decides to sell information to the highest bidder; this might be preceded by certain actions that could be detected. In this thesis we have implemented a model for detection of insider threats adapted for companies which is usable when trying to reduce the risk of insider threats.

Detection of insider threats can be done with the help of machine learning. Machine learning is when a computer learns from input data without being specifically programmed. In our case we use a machine learning algorithm, called Isolation Forest, which is specialized in detecting anomalies. Machine learning typically needs a large amount of data to be able to perform well. This leads us to a common problem among researchers of insider threats - real data is often not made public. Most companies treat data of internal attacks, insider threats, with high confidentially, and do not make it publicly available. This led a team of researchers to develop a synthetic data set, adapted for researches who research about insider threats. The data set consists of lots of normal employee behavior as well as a small part of suspicious events that indicates insider threats. The small part of only only 0.023% suspicious events introduces problems that several machine learning algorithms have a hard time to handle, but anomaly detection algorithms such as Isolation Forest can deal with it quite well.

Inspired by the current discourse with legislations related to privacy and integrity for citizens of the EU (General Data Protection Regulation, EU ePrivacy Regulation) we have decided to restrict our data usage to monitor data that is less of a privacy concern for a company's employees. Specifically login times and logout times at office computers.

Preprocessing of the input data is an important aspect of machine learning. To be able to get as good results as possible from the machine learning algorithm, one need to have

adapted the raw data for the algorithm. Different methods of preprocessing gives large differences in accuracy of the machine learning model. By evaluating different ways of preprocessing our data set, we could conclude that periodical features where better than ordinal features.

In this thesis we have concluded that by designing a model for detecting insider threats using arrival and departure times, we were able to detect 76% of all insider threats while only falsely classify 7% of all normal events as threats. Although the false positives can be expensive to handle in terms of manpower and further analysis, the detection rate of 76% could potentially save a company from a otherwise very expensive data breach.

Chapter 1
Introduction

Companies face a common problem: they need employees but their employees can also be a potential threat to the company. This is a problem that we call *insider threat*. To prevent the problem, a company usually secure its assets using security policies. But a security policy is a blunt weapon against insider threats and is often overlooked, either intentionally or unintentionally, by employees [13].

What if we could detect and prevent insider threats before they ever happen? For example, if an employee feels let down by the company and decides to sell information to the highest bidder; this might be preceded by certain actions that could be detected. In turn, the threat could be avoided by involving the human resources department to talk to the individual and resolve the conflict before the fault ever appeared. Simply put, a preemptive approach could prevent the damage from ever happening. But the approach needs to be respectful and cost-effective, to prevent further losses. With this idea in mind we want to evaluate the possibilities to develop a system that can detect insider threats without violating employees' integrity.

1.1 Problem definition

Since a security breach initiated by an employee can have severe consequences for a company, we believe that there is much to gain by implementing control systems that monitor employees in order to detect and prevent such breaches. At the same time companies need to ensure individual's privacy and integrity. With the pressure from new regulations such as the General Data Protection Regulation (GDPR) [1] and the anticipated EU ePrivacy Regulation (EPR) [2] to ensure integrity on a personal level, it is interesting to know how well insider threats can be detected using as little information as possible. To balance between an effective system and integrity we decided to analyze how many insider threats that can be detected by only analyzing the log on and log off times. Analyzing log on, log off times could be compared to analyzing when an employee is present at the workplace.

1.2 Objective

The purpose of our data analysis is to determine what behavior is relevant to discover who poses an insider threat. We will propose a method that provides real time information about potential threats and evaluate different methods of processing the input data in order to achieve accurate predictions. We will discuss different angles of evaluating the results in terms off different evaluation metrics and what they may correspond to in a real life scenario. Below are the key objectives that fall into the scope of this thesis.

- Make an exploratory analysis of the CERT [1] insider threat data set.
- Preprocess data in order to improve the quality of our proposed algorithm
- Extract relevant features for insider threat analysis
- Apply the Isolation Forest Machine learning algorithm to the data set
- Analyze which features that are most important to get a sufficiently reliable result
- Evaluation of how these results can be used to increase security and reduce profit loss in a company

1.3 Limitations

To narrow down the scope of analysis we have chosen to analyze one Machine Learning algorithm. We have chosen to focus on unsupervised learning and anomaly detection since it is well suited for anomaly detection. When the data is sparse, with few true positives, it is hard to train a model using traditional supervised learning approaches [23]. After considering several algorithms including among others: Isolation Forest, Random Forest, SVM, K-nearest neighbor and K-means clustering; we choose to limit our scope to Isolation Forest. This is motivated by Isolation Forests characteristics. Isolation Forest shows promising results using sparse data sets and works well in high dimensional problems which have a large number of irrelevant attributes, and in situations where the training set does not contain any anomalies [18].

1.3.1 Synthetic Data

To be able to do thorough research that prevents insider threats, data is needed, but it is difficult to obtain real insider threat data. An organization would need to monitor and record the actions of its employees. This raises integrity and privacy concerns. One major limitation is the data needed for analysis and to avoid this, synthetic data is often used for research purposes [12]. This is obviously a limiting factor.

In order to facilitate research on insider threats, CERT [12] has created a data set that is open access. We have chosen to use this data. The synthetic data generated by CERT

[1] CERT (Computer Emergency Response Team) division at the Software Engineering Institute at Carnegie Mellon University, in partnership with ExactData LLC, and under sponsorship from DARPA's Information Innovation Office [12]

mimics real insider threat data and is developed specifically for research about insider threats.

Fully synthetic data can not replace real data, but it proves useful, especially for testing confirmatory hypotheses [12]. Several research papers use synthetic data to conclude if their schemes perform well [4][18], which led us to consider synthetic data to be sufficient for our thesis.

1.3.2 Data encoding

Machine Learning algorithms can be expensive in terms of computational power and memory requirements. The data set used in this paper includes two categorical features that consist of 1000 unique categories each. Certain encoding methods, like the so-called one-hot-encoding method, can be very useful in machine learning, but also very costly. To perfectly encode the two mentioned features using this method requires a large amount of memory as it would result in 2000 new features. This has somewhat limited our evaluation using this encoding method. Another encoding method, which does not have this limitation, is the Label Encoder. Using this encoding method we can run the algorithms quickly and efficiently. However Label Encoding introduces another unwanted property as it causes the machine learning algorithm interpret the features as if they are in some inherent order.

1.3.3 Randomness

In this thesis we have evaluated some methods of data-preprocessing in order to see if they contribute to the isolation of the threat-labeled data points. The evaluation has been performed by running the Isolation Forest algorithm on all combinations of features and then calculating the two-tailed Kolmogorov-Smirnov score between threat-labeled and non-threat-labeled data points. This method is effective in giving an indication to which preprocessing method of the input data that gives us a better isolation of the threat-labeled data. However, since there is a random component to the Isolation Forest algorithm each time one run the Isolation Forest algorithm one will end up with slightly different results. Therefore we must run the evaluation several times in order to gather enough data to determine that the different results between runs with different features are statistically significant and not due to the random deviation added by Isolation Forest.

1.4 Contributions

This project contributes to the current research by proposing a scheme indicating how insider threats can be prevented by analyzing information that can be obtained without online surveillance. By focusing on logon data we have a chance of finding a method of insider threat detection that is more easily generalizable to other areas of surveillance. If for example it is not possible to monitor the network and/or computers in use to gain knowledge about email content, website visits or usage of external devices, it might still be possible to monitor activity similar to logging in and out on a PC. In a work environment where bring-your-own-device (BYOD) is employed or when employees are working from home

and connecting to the corporate network using VPN, the logon and logoff activities might still be detectable by logging when a known computer's network interface is activated in the office or when a VPN user is connecting to the network.

As opposed to previous work conducted by Aaron et al. [30] we provide insider threat detection in real time. Rather than aggregating data over time and analyzing it retroactively on a per-day basis we analyze the data on a per-event basis. Aaron et al. use an online surveillance approach to detecting insider threats, using a similar data set as we do. However they use the whole v6.2 data set, while we use only the logon file from the v4.2 data set. Since we limit ourselves to analyzing information that is socially acceptable to use (only when you are working), compared to using the complete data set of another version, our results are not directly comparable.

We seek to find an efficient way to find insider threats in real time. In order to do this we must minimize the dimensions of the data in order to do fast processing while still keeping as much of the entropy as possible in order to detect threats with high accuracy. Furthermore, we aim to protect the individuals Privacy and Integrity which we consider important when GDPR and the anticipated ERP is coming into place.

Chapter 2
Related Research

In this chapter we introduce the data set upon which we base our research. We discuss research related to insider threat detection and what impact insider threats may have on a company. To determine if a company should invest in insider threat risk reduction is a hard decision to make. To begin with, how will an insider threat that is carried out impact the company, and what drives a person to commit fraud? Research about these aspects are discussed in Section 2.1. After determining that there exists a need for insider threat detection, how to detect insider threats using machine learning is analyzed in the following sections. Isolation Forest is one unsupervised machine learning algorithm adapted for sparse data sets, which is useful for insider threat detection where the number of positives are low.

Lastly, an insider detection scheme using online surveillance is presented. For companies prepared to perform online surveillance on their employees, their scheme is suitable.

2.1 Insider threats

Our objective is to reduce the risk of insider threats. Therefore the insider threat needs to be defined. Asim Majeed et al. [19] have defined insider threats and what potential impact they may inflict on a company. As more and more companies are embracing BYOD, where employees work on their own computers and devices that the company has no control over, the risk of insider threats is increasing. An employee posing an insider threat may do it both intentionally and unintentionally. An inside attack can be severe, and descriptions of some possible scenarios are shown in Table 2.1, which shows the information, potential leak, unintended visibility to outsiders and its impact on the enterprise. Threats from within organizations shall definitely not be overlooked when considering digital theft and cybercrime. According to a survey by the US Justice Department [24] 40% of all computer security offenses reported where suspected to involve insiders.

As an explanation for why an employee might commit fraud; and thereby pose an in-

sider threat, we have analyzed the research by Wolfe and Hermanson [31]. They propose that the motivations for committing fraud are based on four conditions, which can be drawn in a so called fraud diamond, see Figure 2.1. The fraud diamond was proposed after analyzing why employees are inclined to commit fraud. There are four conditions that need to be present for fraud to occur — pressure, opportunity, rationalisation and capability. In our data set, we can see in the scenario description in Section 2.2.1 that all of the scenarios shows signs of Pressure, Opportunity, Ability and Rationalisation. According to the fraud diamond, this leads to fraud.

Also Solomon Mekonnen et al. [20] use the fraud diamond. They present a privacy preserving context-aware insider threat prediction and prevention model where the prediction and prevention is based on the components of the fraud diamond. To get a broader incite to insider threat the reader is referred to [10] where insider threats are thoroughly analyzed from a corporate and commercial angle as well as from the view point of a state or government.

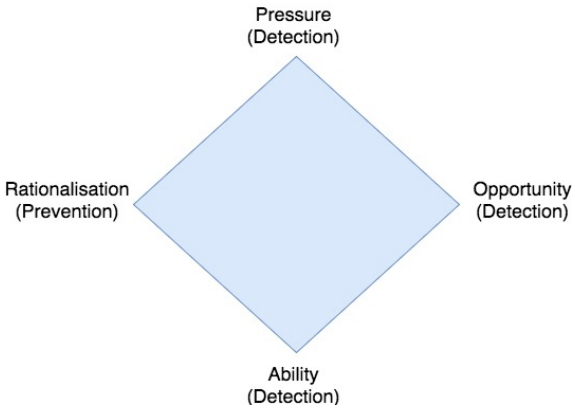

Figure 2.1: The Fraud Diamond [31], shows which conditions that are needed for fraud to occur: Pressure, Opportunity, Ability and Rationalisation.

Abdulaziz Almehmadi and Khalil El-Khatib [5] have presented a futuristic approach to how insider threats can be prevented, since Machine Learning is not the only approach. Abdulaziz Almehmadi and Khalil El-Khatib [5] claim that all existing access control systems rely on identity authentication on design, which means that they are not suitable for insider threat detection. Since there is no continuous evaluation of the user's level of trust, the high risk of insider threat remains the most common threat to an organization. Abdulaziz Almehmadi and Khalil El-Khatib [5] have developed a scheme using Intent Based Access Control (IBAC) that uses brain signals as an intention access control. The IBAC scheme is a non identity-based access control as it requires knowledge only of why access is being requested, not of who is requesting access. [5]

2.2 Data set description

The CERT division at the Software Engineering Institute at Carnegie Mellon University, in partnership with ExactData LLC, and under sponsorship from DARPA's Information

Table 2.1: Insider threats table from Asim Majeed et al. [19]

Insider Threats			
Information	**Potential Leak**	**Unintended Visibility to Outsiders**	**Impact**
Trade Secrets	Stored on the network and shared drive	Competitors	Company Defamatio
Financial Details	Copied on external removable media device	Regulators	Monetary Expense for each record lost
HR and Personnel details	Transmitted electronically e.g. email	Unauthorized internal users	Legal Liabilities
Intellectual Property		Press or Media	Loss of Assets
Personal Health Records			Breach of Customer and Clients Trust
Management/planning and Strategic Leaks			Closing of the Business

Innovation Office has created the data set used in this thesis [12]. The data set is entirely synthetic. Both the background data and the data representing malicious actors are synthetically produced by their software.

Some "dirty data" — logical errors that could not possibly happen in real life — are deliberately inserted into the data set. However, it is likely that real world data would contain a higher degree of dirty data due to software exceptions, network problems, untimely shut downs of systems etc.

The data set used is the 4.2 data set. It is a so-called dense needle, which means that it contains an unusually large number of insider threats compared to the other data sets by CERT. This means that we can get more confident results. The other data sets contain too few insider threats to be able to evaluate our model.

The files in the 4.2 data set are as follows:

- `logon.csv`
- `device.csv`
- `email.csv`
- `file.csv`
- `http.csv`
- `psycometric.csv`
- `LDAP` - folder with 18 files, one for each month.

The description of these files is found in the readme included in the data set, and is

attached in Appendix C, the description of the `logon.csv` file is described in Figure 2.2. We focus on the `logon.csv` file, which contains 854 859 lines, where the first line contains the fields `id, date, user, pc and activity (Logon/Logoff)`.

```
logon.csv
* Fields: id, date, user, pc, activity (Logon/Logoff)
* Weekends and statutory holidays (but not personal vacations)
are included as days when fewer people work.
* No user may log onto a machine where another user is already logged on,
unless the first user has locked the screen.
* Logoff requires preceding logon
* A small number of daily logons are intentionally not recorded to simulate dirty data.
* Some logons occur after-hours
  - After-hours logins and after-hours thumb drive usage are intended to be significant.
* Logons precede other PC activity
* Screen unlocks are recorded as logons. Screen locks are not recorded.
* Any particular user's average habits persist day-to-day
  - Start time (+ a small amount of variance)
  - Length of work day (+ a small amount of variance)
  - After-hours work: some users will logon after-hours, most will not
* Some employees leave the organization:
no new logon activity from the default start time on the day of termination
* 1k users, each with an assigned PC
* 100 shared machines used by some of the users in addition to their assigned PC.
These are shared in the sense of a computer lab, not in the sense of a
Unix server or Windows Terminal Server.
* Systems administrators with global access privileges
are identified by job role "ITAdmin".
* Some users log into another user's dedicated machine from time to time.
```

Figure 2.2: The `readme` of the `logon.csv`

The data set only contains data about how the company's employees are acting; the data about who is an insider threat is found in the answers directory. The malicious activity data is found in the answer key file, where there is also data about which insider threat scenario is relevant for each of the data sets as well as data about each of the insiders, so their actions can be analyzed.

2.2.1 Brief Scenario Descriptions

The data set contains a description of the so-called red alert scenarios. These scenarios are series of events that are typically found when an insider threat is apparent. The data set is generated based on these red alert scenarios and every event found as a row in the files in the data set (such as a row in the `logon.csv` file) that is generated in correlation to these red alert scenarios is defined as a threat; and we mark them as a insider threat while evaluating our model.

The description is as follows [12].

1. [A] User who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to wikileaks.org. [The user] Leaves the organization shortly thereafter.
2. [A] User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.

3. [A] System administrator becomes disgruntled. [She] Downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

4. A user logs into another user's machine and searches for interesting files, emailing to their home email. This behavior occurs more and more frequently over a 3 month period.

5. A member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain.

2.3 Isolation Forest

Isolation Forest is a machine learning anomaly detection algorithm created by Fei Tony Liu et al. [18]. According to them, most existing model-based approaches to anomaly detection construct a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies. Isolation Forest on the other hand uses a different approach that explicitly isolates anomalies instead of profiling normal points. Isolation Forest isolates anomalies using their quantitative properties: they are few and different. They have shown that a tree structure can be constructed effectively to isolate every single instance. In multiple dimensions this is done by randomly constructing hyperplanes that separates instances from one another. The number of hyperplanes needed in order to separate out a single instance corresponds to the instances' anomaly score. Below we try to exemplify how this would work in two dimensions, where the instances are separated by lines. Illustration can be found in Figure 2.3.

1. A line is randomly drawn through the collection of instances.

2. An orthogonal line is randomly drawn.

3. The above process continues until all instances are isolated.

4. The iTree is built based on the instances. Each Instance is a leaf in the iTree, see Figure 2.4. Each line represent a node, in other words a decision point where the choice is made if the specific instance is larger or smaller than the value.

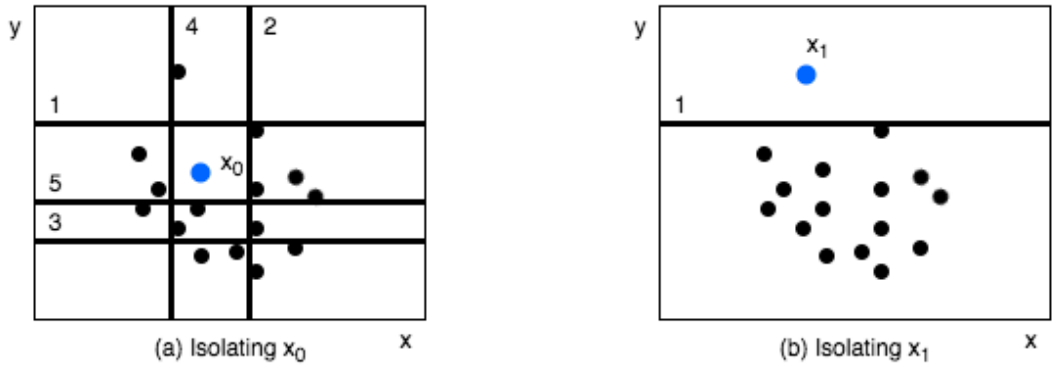

Figure 2.3: Steps for isolating instances. A normal instance (a) x_0 have an isolation path of five, and an anomaly (b) x_1 has a shorter isolation path of two.

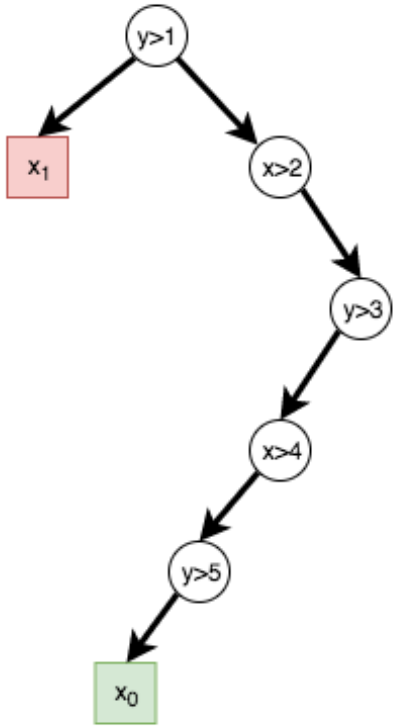

Figure 2.4: Example of the beginning of an Isolation tree, only containing x_0 and x_1, isolating x_1(Anomaly instance) only takes one step, which is found closer to the root of the tree compared to the normal instance x_0 found further away. The numbers represents the lines in Figure 2.3

Isolation Forest locates anomalies effectively since they are close to the root of the tree. It works well in high dimensional problems which have a large number of irrelevant

attributes and in situations where the data set does not contain any anomalies [18]. This makes Isolation Forest a good choice for our data set where the data is biased, with very few true positives, only 0.23‰. Isolation Forest can also be used using only normal data, without any significant loss of performance [18], which is very useful in insider detection [18] since one never know if there exists positives in the data set when it is biased.

2.4 Detecting anomalous user behavior using Isolation Forest

Li Sun et al. [28] use an extended version of the Isolation Forest algorithm to detect anomalous user behavior. They use the Isolation Forest algorithm with a naive approach to extend the algorithm to use categorical features. Their method requires that the categorical data have an ordering, which may be arbitrary. Our results show that this approach can be improved by transforming the categorical features into periodic features.

They use a data set called CA RiskMinder user payroll access logs which consists of 25,991,654 logs, covering a total of 89 days over 14 weeks. Their approach of focusing on 5 out of 42 features and selecting users with an access frequency of 501-600 and building a model for only these users is well suited for the CA RiskMinder user payroll access logs data set. We also use a similar but different approach when we choose to focus on the logon.csv file in the data set from CERT. We do not remove users from the data set.

2.5 Related Research using the CERT data set

Aaron et al. [30] have published a paper on a similar subject with a similar approach, and they have also used the CERT data set. They use features which require that you constantly monitor employees' computers and network activity, while we find that approach infeasible for companies due to privacy and integrity concerns. We do not seek to monitor online activities, but rather activities commonly accessible such as arrival and leaving times for the employee.

They [30] use the 6.2 version of the data set and present an online, unsupervised deep learning approach to detect anomalous network activity from system logs in real time. When they extract features they categorize them into two categories: categorical user attribute features and continuous "count" features. Categorical user features refer to attributes such as a user's role, department, and supervisor in the organization. Continuous "count" features refers to, for example, the number of uncommon non-decoy file copies from removable media between the hours of 12:00 p.m. and 6:00 p.m.

Aaron et al. [30] assess the effect of including or excluding the categorical variables in their models. They conclude that their model performs better without the categorical variables that were given to provide context to the model. This led them to only use count features for most of their experiments. An important detail is that Aaron et al. [30] aggregate the events to so called user days, and they only analyze weekdays and not weekends.

They use five sources of events: logon/logoff activity, http traffic, file operations, and external storage device usage.

As explained in Chapter 4, we do not aggregate the events to the so-called user days, but we rather analyze the events on their own. Therefore the event of a login action at 3 am on a weekend could raise suspicion but the same user's action of login at noon the same day may not raise suspicion, since the actions are not aggregated. We analyze both weekdays and weekends. Furthermore, we find that http traffic, file operations and external device usage is unfeasible when trying to protect the employee's privacy and integrity. The three previously mentioned sources all demands installation of surveillance software, which is not combinable with BYOD.

Cumulative Recall k, (CR-k) is used by Aaron et al. [30] as a tuning criteria. CR-k can be thought of as an approximation of the area of the recall curve. The recall curve is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.1}$$

(where TP is true positive and FN is false negatives). Inspired by Aaron et al. [30], we use recall as an evaluation metric. Recall is well fit for biased data set since it does not take the large amount of negatives into consideration.

2.6 Conclusion

As seen in Section 2.1 it is important that companies considers the insider threat when they are evaluating how to secure their assets. A security breach initiated from within a company could potentially be expensive. This is obviously an incitement for companies to put control mechanisms in place that could detect and prevent these breaches. To evaluate how this is best done one has to consider what data is technically available as well which part of that could be used when taking the privacy and integrity of the employees into account.

Related research suggests that Isolation Forest is a suitable algorithm for classifying anomalies. We will use this algorithm in order to classify the events in the CERT data set either threats or non-threats as effectively as possible by examining the not so intrusive information of employees logging in and out of computers in an office environment.

Chapter 3
Theory

3.1 Machine Learning

Machine learning is a field of computer science that uses statistical mathematical techniques to make predictions and estimations or to identify certain patterns from data without being explicitly programmed. A widely accepted definition is the definition by Tom M. Mitchell [21]. He formally defines machine learning as follows:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"*

Machine learning is a broad concept and is applied in several areas, such as identification of objects in images, determining diseases and anomaly detection. Machine learning is used to estimate the outcome of a situation. Such outcomes are divided into two groups: quantitative (such as prices) and categorical (such as cats and dogs).

3.1.1 Categorical Classifiers

Machine learning can be used to solve so-called classification problems. Here the problem is that the algorithm needs to determine the category of the data, where the possible categories are included in a discrete set, as described by Tom M. Mitchell [21]. The simplest kind of categorical classifier is the binary classifier. The binary classifier is used in situations where the answer is always one out of two options, commonly seen in yes/no classifications. If the machine learning problem is to determine if an animal is a dog or not, one can use the binary classifier. On the other hand, if the problem is to determine what kind of animal the animal is - dog, cat, mouse and so fourth, one cannot use the binary classifier. This thesis will focus on binary classifiers since determining whether an employee poses an insider threat or not is a binary classification problem.

3.1.2 Supervised Learning

Supervised learning is when the model is trained on a training set and then the trained model is used to predict results in a machine learning problem, either quantitative or categorical. Labeled training data is needed to use supervised learning. Examples of supervised learning algorithms are introduced in Table 3.1. Supervised learning algorithms require a potentially expensive training process and are therefore obstructed by a typically small quantity of insider threat data available for training [23].

While choosing the best algorithm for the problem, four major issues should be taken into account: Bias-Variance Trade-off, Function Complexity and Amount of Training Data, Dimensionality of the Input Space and Noise in the Output Values; these are further explained below. By carefully analyzing these issues, choosing an effective algorithm can be simplified.

Bias-Variance Trade-off

While choosing the algorithm, two sources of error need to be minimized at the same time. One is bias and the other is variance. The bias is an error resulting from a wrong assumption in the learning algorithm. High bias can cause the algorithm to underfit meaning and thus miss relevant relations between features and target outputs. The variance is an error resulting from sensitivity to small fluctuations in the training set. High variance can cause the algorithm to overfit by modeling noise in the training data. This leads to a dilemma, since minimizing the bias will increase the variance error and vice versa [14]. Therefor a trade-off has to be made, in order to choose the best algorithm for the current case.

Function Complexity and Amount of Training Data

A simple function can be easily determined by a learning algorithm with high bias and low variance from a small set of data. But if the function is highly complex the best approach is to use an algorithm with low bias and high variance as well as a large set of training data [7]. This problem is often found in supervised learning for Insider Threat detection scenarios, such as in the research by Pallabi Patveen et al. [23].

Dimensionality of the Input Space

If the input vector has a high dimension, the learning problem can be difficult even if the result depends on only a few of the features. This can give an error since the model trains on features which are not relevant to the outcome. This causes confusion for the algorithm, leading to high variance. This error can be reduced by proper data preprocessing such as manually removing irrelevant features. Furthermore, there are many algorithms for feature selection, which have the goal of identifying the most relevant features [4]. Dimensionally reduction, in the sense of finding out the most relevant features to detect insider threats, is an objective of this thesis.

Noise in the Output Values

As a fourth aspect, the acceptable degree of noise in the output is important. If the desired output values are often incorrect, the method should not attempt to find a function that exactly matches the training data. Attempting to fit the data too tightly to the training data leads to overfitting. It is possible to overfit both if there are stochastic errors and if there is a too complex function that the model is trying to mimic [25].

Table 3.1: Supervised machine learning algorithms from Chunxiao Jiang et al. [15].

Category	Learning techniques	Key characteristics
Supervised Learning	Regression models	Estimate the variables' relationships Linear and logistics regression
	K-nearest neighbor	Majority vote of neighbors
	Support Vector Machines	Non-linear mapping to high dimension Separate hyperplane classification
	Bayesian learning	A posteriori distribution calculation

3.1.3 Unsupervised Learning

Unsupervised learning is the machine learning task of analyzing data that does not contain labels in order to receive a prediction. Unsupervised algorithms cannot learn from previous data, in the sense of a given behavior leading to a given result, and be sure that the result is correct. An unsupervised algorithm uses another approach,to detect data that behaves in a similar fashion. Many unsupervised algorithms are so-called anomaly detection algorithms [6]. Examples of unsupervised learning algorithms are introduced in Table 3.2. We have considered these algorithms in regard to biased anomaly detection, and our conclusion was that although not mentioned by Chunxiao Jiang et al. [15], Isolation Forest were better adapted to our objective.

Table 3.2: Unsupervised machine learning algorithms from Chunxiao Jiang et al. [15].

Category	Learning techniques	Key characteristics
Unsupervised Learning	K-means clustering	K partition clustering Iterative updating algorithm
	PCA	Orthogonal transformation
	ICA	Reveal hidden independent factors

3.1.4 Anomaly Detection

Anomaly detection algorithms are often used in intrusion detection, fraud detection and fault detection. These use cases typically have very unbalanced data sets, which necessitates a different approach than those used on balanced data sets to achieve reliable results. Anomaly detection algorithms can be implemented using different types of models:

unsupervised learning, supervised learning and semi-supervised learning [28]. We have chosen to use Isolation Forest which is an example of a unsupervised learning approach for anomaly detection.

Unsupervised Learning

Unsupervised learning algorithms detect anomalies in unlabeled test data sets by assuming that the majority of the events in the data set are normal. The algorithm is looking for events that do not fit the majority of the events and labels them as anomalies. The problem with this approach is that the algorithm can only detect anomalies that conform with its model, due to lack of anomaly instances [28]. Isolation Forest fits into this category of anomaly detection models.

Supervised Learning

Supervised learning techniques require that the training data set is labeled in a binary way with what is normal and which events that are anomalies. Many algorithms not adapted for anomaly detection have trouble doing this, due to the unbalanced nature of anomalies. Examples of anomaly detecting using supervised learning include support vector machines [29] and Bayesian networks [8][28].

Semi-Supervised Learning

Semi-Supervised learning methods construct a model representing normal behavior from a normal data set. Then the method predicts if the test data is likely to have been generated by the learned model [28]. Using these models, a normal data set labeled can be used first, and with continued training on a unlabeled data set, the model can learn from this too by recognizing similarities between instances present in both of the data sets.

3.2 Evaluation of Machine Learning Models

To be able to evaluate how well a machine learning model performs, standard metrics are used. We use these metrics to evaluate how well our model performed, and by using the same standard metrics researchers can compare their results. The relevant theory for these evaluation techniques are presented in this section.

3.2.1 Kolmogorov–Smirnov test

The Kolmogorov-Smirnov test is a non-parametric test for measuring the strength of a hypothesis that some data is drawn from a fixed distribution (one-sample test), or that two sets of data are drawn from the same distribution (two-sample test) [17]. We use the two-sampled Kolmogorov-Smirnov test, henceforth referred to as KS-test, to determine if an event is likely to be a normal event, i.e. non-threat or an anomalous event, that is an insider threat.

3.2.2 Confusion matrix

The confusion matrix is commonly used to determine how well a classification model performs [28]. Classification results belonging to two classes, True and False, are listed in a 2x2 confusion matrix, see Table 3.3. By analyzing the number of True Negative (TN), False Negative (FN), False Positive (FP) and True Positive (TP) we can determine the effectiveness of the approach.

Table 3.3: Confusion matrix with insider threats and normal behavior mapped as the categories.

Predicted / Actual	Normal	Insider Threat
Normal	TN = True Negative	FP = False Positive
Insider Threat	FN = False Negative	TP = True Positive

3.2.3 Matthews Correlation Coefficient

For weighted binary classification problems, it is recommended by Davide Chicco [9] to evaluate the algorithm's performance using Matthews Correlation Coefficient (MCC). Commonly, two other scores are used to determine how well a model is performing, *accuracy* and *F1 score* [9], but these are often misleading [9] since they do not fully consider the size of the four classes of the confusion matrix in their final score computation. MCC takes into account the balance ratios of the four classes of the confusion matrix. The MCC value takes both *accuracy* and *F1 score* into consideration which is parameters one want to optimize to find a harmonic mean where detection rate is maximized and the false detection rate is minimized. Using the MCC value one can avoid to tune the ML model to detect all positives and thereby introducing a large error rate.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{3.1}$$

Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. In our case, it is the relation of correctly identified threats and normal situations, to the total number of all cases examined. The result of this metric may be misleading in cases like ours where the data is heavily unbalanced. Identifying every situation as a normal situation would produce a result very close to 1, but would also make the result meaningless.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{0 + largenumber}{Totalnumberindataset} \approx 1 \tag{3.2}$$

3.2.4 F1 score

The F1 score is the harmonic mean of precision and recall and is used as a measure of a test's accuracy. The F score can be weighted to emphasis whichever of the precision and recall parameters is deemed the most important, if the F score is weighted to its harmonic

mean it is called the F1 score. If recall is more important, the F2 score can be used, the F2 score weighs recall higher than precision.

$$F_1Score = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = \frac{2}{\frac{1}{Recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.3}$$

3.2.5 Precision Recall

Since we are using a sparse data set with many negative instances and very few positive instances, it is hard to find any evaluation metrics that fit our data set. MCC takes all of the four metrics in the confusion matrix under consideration, which in our case can be misleading. Our insider threat detection problem bears similarities to computational biology problems, in that it has many negative instances and very few positive instances. For these kinds of problems, it is wise to avoid the involvement of TN (True Negatives), since this will weight the problem negatively [9].

Precision

In the classification problem, precision shows how many instances were correctly identified in the positively identified set. Since we seek to identify insider threats whilst keeping the false positives low to reduce cost, this is an important metric.

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

Recall

Recall shows how complete the results are. This is done by taking the correctly identified insider threats in relation to all of the insider threats. The higher recall, the more threats the model has been able to identify. Recall is a good metrics in biased data sets since it does not take any negatives into account. It is not fit as a tuning criteria for the same reason, only taking recall into consideration while tuning the model can give misleading results, since classifying everything as a positive would maximize the recall. We use recall as an evaluation metric.

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

3.3 Conclusion

What machine learning algorithm that is suitable for a classification problem depends on the input data. Also the method when it comes to evaluating the quality of results is largely dependent on what kind of data is analyzed. When suitable learning algorithms and evaluation methods are chosen a lot of focus needs to go in to processing the input data so that it can be used for classification as effectively as possible.

After running the machine learning model, it is also important to be able to evaluate the model. The model needs to be tuned taking several parameters into consideration, leading us to choose Matthews Correlation Coefficient as a tuning criteria, since it represents a harmonic mean where the detection rate is maximized and the false detection rate is minimized. For evaluation of biased data sets after tuning recall is a better evaluation metric.

Chapter 4
Approach

4.1 Approach summary

Our approach can be summarized by the following steps:

1. Choose a relevant machine learning algorithm for prediction
2. Extract features, data preprocessing
3. Do feature optimization i.e analyze which features that are most important
4. Use these to determine at which confidence level an inside threat can be predicted
5. Analyze the prediction results using evaluation metrics

4.2 Choice of Machine Learning Algorithm

The data set we use contains only 0.23‰ positives, which makes it a sparse data set. This introduces complexity in that we need to choose an algorithm appropriate to sparse data sets. Many supervised machine learning algorithms do not perform well on sparse data sets since such data sets have too little training data [4][23]. After considering the capabilities of supervised and unsupervised machine learning algorithms, described in Section 3.1, we concluded that an unsupervised approach would be a better fit for our problem.

Unsupervised algorithms have the advantage that they do not need to be trained, which also opens up the door for smaller data sets. There are many different unsupervised algorithms but by reviewing several papers [18][15][30][28] and searching for an algorithm that could handle our classification problem using a sparse data set, we found Isolation Forest. These papers use sparse data sets and Isolation Forest in combination with good results, and averagely it seemed like the best standardized algorithm for the usage with

sparse data sets. Furthermore, Isolation Forest has been proven efficient in prediction using sparse data sets [18]. This led us to choose Isolation Forest as a basis for our thesis, with the possibility of improvement of the thesis by comparing with other algorithms.

4.3 Data preprocessing

The `logon.csv` file from the data set from CERT [12] contains five parameters and they are generated in a way that is humanly understandable, see Table 4.1. Thus comes the need for preprocessing and feature extraction in order to be able to use the data with machine learning algorithms. In our case, the data preprocessing is essentially about converting the human readable data in our data set to a form that is more useful for the machine learning algorithm. Preprocessing is often a time consuming but important part of machine learning.

4.3.1 Preprocessing

We use the data set from CERT [12], and focus specifically on the`logon.csv` file. The file is formatted as in Table 4.1.The data in the file needs to be preprocessed and changed into a form that a machine learning algorithm can make sense out of. In particular, if the datetime stamp in the second column were to be interpreted just as a string, the algorithm would not find any meaningful relationships between different datetime stamps.

Converting the features to ordinal integer data, e.g. number of seconds since 1st of January 1970, would make them comparable relative to each other but a lot of information would be lost. We estimated that information such as month, day of month, hour of the day and day of the week could be significant information that is possible to extract from a datetime stamp. By parsing the CSV file line by line using Python and its `datetime` module as well as some string manipulation, the datetime feature in the data set was split up into three new features: weekday, month, and hour. To avoid noise we choose to exclude the minutes.

The `id` column of the CSV file was also used in order to create the threat column. By searching for the id in the answers file the threat column was populated with a binary representation of the answer; a '1' indicates that the row represents an insider threat and a '0' indicates that it does not. After processing we obtained a new CSV file that could more easily be interpreted by machine learning algorithms. The new file content is shown in Table 4.2.

Table 4.1: Format of the Login.csv file.

id	datetime	user	pc	activity
{X1D9-S0ES98JV-5357PWMI}	01/02/2010 06:49:00	NGF0157	PC-6056	Logon
{G2B3-L6EJ61GT-2222RKSO}	01/02/2010 06:50:00	LRR0148	PC-4275	Logon
{U6Q3-U0WE70UA-3770UREL}	01/02/2010 06:53:04	LRR0148	PC-4124	Logon

Table 4.2: Format of the transformed file.

user	pc	activity	weekday	hour	month	threat
NGF0157	PC-6056	Logon	0	6	2	0
LRR0148	PC-4275	Logon	0	6	2	0
LRR0148	PC-4124	Logon	0	6	2	0

4.3.2 Handling Categorical Features

We are using the `sklearn.preprocessing.LabelEncoder` library [27] in order to encode the categorical features in our data set. This encoding method translates each unique category into an integer value that can be used by the isolation forest algorithm.

We have followed the method for feature pre-processing described in the paper written by Aburomman and Reaz [4]. Aburomman and Reaz [4] use the KDD99 data set which exhibits similar difficulties to the data set from CERT that we are using. The KDD99 data set contains some categorical features such as protocol type, service and flag. These categorical features are converted using so-called one hot encoding. This encoding method creates a new feature — a new column in the data set — for each unique category in a feature. For this we are using the Python library `pandas.get_dummies` [22]. One hot encoding on our data set is evaluated in Section 5.4.

4.3.3 Periodic Features

In [16] David Kaleko explains that as an alternative to handling features such as time with integers 0-23, where 23 and 0 are very far apart, as seen in Figure 4.1, one could convert the features into a periodic form. The numerical features that can be produced by preprocessing the datestamp in our data set are all ordinal in their simplest representation. After hour 14 comes hour 15, after Monday (day 0) comes Tuesday (day 1). However using this representation we lose the information that after hour 23 comes hour 0 and after Sunday comes Monday. In an attempt to solve this problem we converted each of these features into two coordinates on the unit circle (see Figure 4.2 and Figure 4.3). By representing the coordinates with their sinusoidal and co-sinusoidal values respectively, we have added the sought-after attribute of periodicity. Table 4.3 shows how the data set looks after encoding the categorical features and transforming the periodic features.

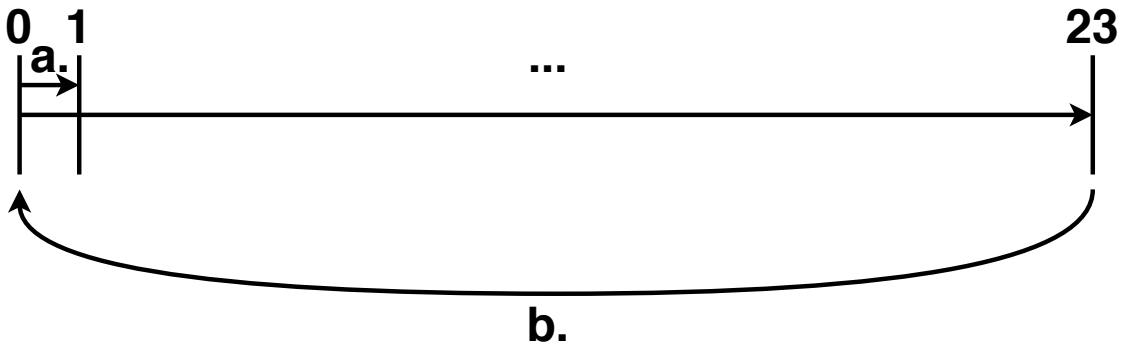

Figure 4.1: Ordinal representation of time. The distance between the data points in a. is much smaller than in b.

$$w_{sin} = sin(\frac{2\pi w_i}{7}) \tag{4.1}$$

$$w_{cos} = cos(\frac{2\pi w_i}{7}) \tag{4.2}$$

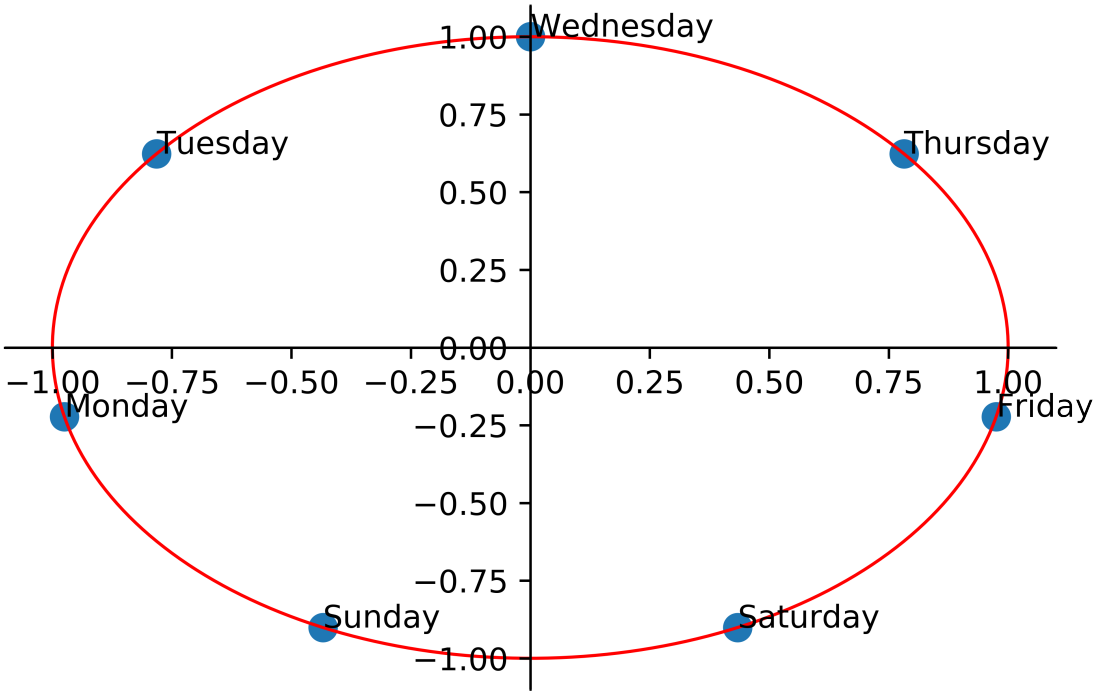

Figure 4.2: Periodic representation of weekdays

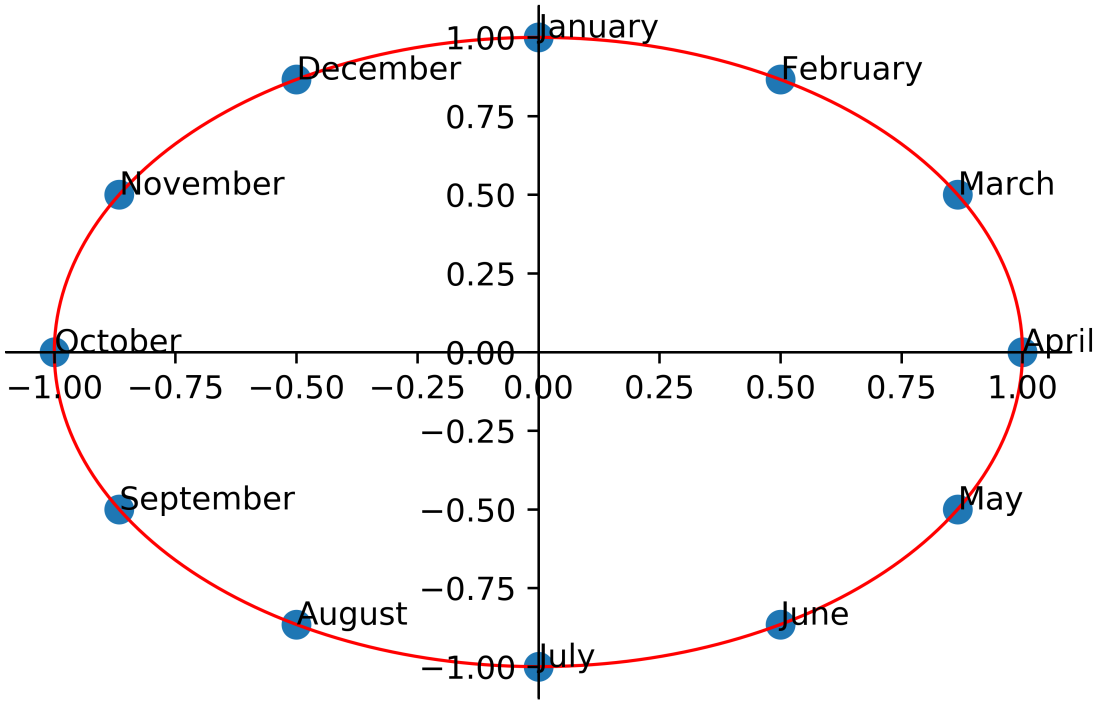

Figure 4.3: Periodic representation of months

Table 4.3: Format of the transformed file with periodic features

Parameters	Event 1	Event 2	Event 3
user	707	620	620
pc	589	425	410
activity	1	1	1
threat	0	0	0
hour_sin	1	1	1
hour_cos	6.123234E-17	6.123234E-17	6.123234E-17
month_sin	0	0	0
month_cos	1	1	1
weekday_sin	0.866025	0.866025	0.866025
weekday_cos	-0.5	-0.5	-0.5

4.4 Feature optimization

After preprocessing the data another problem with the data arises: the need to optimize the data according to the information content in different features. We do this by making a KS-evaluation of every possible combination of the logon data set. This to determine which features contain the most information value. The results for this evaluation can be found in Appendix B.1 and Appendix B.2.

By running the machine learning algorithm multiple times with different combinations of features we can determine which features are most significant in order to determine the

inside threats. We used seven different features, so $n = 6$ (n includes 0), which means that the total number of unique combinations of features is 63, as seen in the formula below.

$$\sum_{k=1}^{n} \binom{n}{k} = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!} = \sum_{k=1}^{6} \frac{6!}{k!(6-k)!} = 63 \tag{4.3}$$

4.5 Evaluation metrics

When we evaluated our results, we used the KS-test to determine if an event is a insider threat or not. Then we used the confusion matrix to calculate quality scores for our results. By maximizing MCC we determined the best limit value for the model. The MCC value will be low because of the high degree of negatives in the data set, but maximizing the MCC value still optimizes our function effectively. Lastly, we use Recall to evaluate the model since the number of negatives does not influence the result, as our data set has only 0.23‰ positives.

4.6 Conclusion

Data preprocessing and feature engineering is a very important part of machine learning. When being limited to using just a few features as input to the classification algorithm even more emphasis needs to be put into extracting high quality features and filter out noisy and/or redundant data. The best preprocessing method is the one that gives us the best results of the classification problem given some method of evaluation.

Chapter 5
Evaluation and discussion

Using the methods of evaluation presented in Section 3.2 we will in this chapter present how we determined what was the most important features in order to solve the classification problem. Also, we evaluate some different ways of doing data preprocessing in the form of encoding methods and feature engineering is evaluated. In order to ensure our results are statistically significant each evaluation process is run multiple times.

5.1 Choosing evaluation metrics

The evaluation metrics for machine learning models, such as those referred to in Table 3.1 and Table 3.2, are often different depending on what problem the machine learning models are trying to solve. Still, the confusion matrix stands as a base through the evaluation of most models. We found that the confusion matrix, on its own, was sufficient to answer simpler questions such as how many threats we identified and at what cost in terms of false positives.

In order to make better use of the data from the confusion matrix, we mainly used Matthews Correlation Constant (MCC) and recall. MCC is a good metric to optimize since it uses both recall and accuracy. When we evaluated some of our test results using MCC, we obtained a very small value, 0.04. In the literature it is mentioned that a value close to 0 means the model is comparable with random guessing, whereas close to 1 means that the model is accurate. At first glance, this seems to indicate that our model performs badly, however this is not the case. Our value becomes small because we used a data set with only 0.23‰ positives. Therefore we chose to evaluate our model with recall instead and to use MCC to find the best limit value. MCC nonetheless shows the best limit value since we seek to maximize both recall and accuracy, but recall better captures the actual performance of our model compared with our objective of identifying potential insider threats.

5.2 Combining features

5.2.1 Preprocessing

Since the algorithm is evaluating the input data on a per-event basis, i.e row by row, it does not have any sense of how long a user stays logged in. This, together with the fact that the time for log in and log out seem to be more relevant to the outcome compared to the other features (pc and user, see Section 5.6) led us to try some feature engineering.

We parsed the data set line by line using Python. If the line was a log-in event we saved the datestamp for this event together with PC and user and started looking for the corresponding logout event. When the logout event was found we calculated the time difference between log in and log out and created a new feature out of this.

5.2.2 Evaluation

The resulting table, shown in Table 5.1 essentially consists of time of log in, time of log out and how long each user has been logged in on a per-PC basis. As stated in the data set description, Section 2.2, there are some inconsistencies in the data set. For example

```
A small number of daily logons are intentionally
not recorded to simulate dirty data.
```

also,

```
Screen unlocks are recorded as logons.
Screen locks are not recorded.
```

This led us to remove some rows of the transformed data set. Specifically we removed rows with a timediff larger than 1440 since it corresponds to 24 hours and it seems unlikely that a real user would be logged in for more than 24 hours. In general we do not want to remove input data in this blunt way but we considered users being logged in for this long a time as outliers that might have been introduced by inconsistencies in the data set and therefore might lead to the model performing worse.

Table 5.1: Table after transformation based on the time difference

weekday_in	month_in	hour_in	weekday_out	month_out	hour_out	timediff
5	01	06	5	01	17	616
5	01	06	5	01	17	617
5	01	06	5	01	07	32
5	01	07	5	01	16	556
5	01	07	5	01	17	617
5	01	07	5	01	18	681
5	01	07	5	01	18	672
5	01	07	5	01	18	669
5	01	07	5	01	17	614

We hypothesized that by using all existing features to create session data and a new feature, timediff, we would improve the accuracy of the model. However, after running the model on the data preprocessed according to this scheme we actually received a higher p-value than the approach with periodic features and no session data. This particular feature extraction method was proved not to be useful. It is possible that the feature that we introduced had no significance when it comes to determining an insider threat and that it therefore just added additional noise. It is also possible that the inconsistencies in the data set, the so called "dirty data" that were deliberately inserted by CERT, had a greater effect on the algorithm after we did the feature combination.

5.3 Label Encoding

The model used, `sklearn.ensemble.IsolationForest`, cannot handle features made up of strings. Therefore we had to encode our categorical features from string values to integers. By doing so the Isolation Forest algorithm could separate the different categories from one another, which is an important characteristic. At the same time, this method of encoding implies an inherent ordering between the categories, which is not true in reality for our categorical features pc, user and activity. Naturally, this is an unwanted property that will have a negative impact on our model.

5.4 One Hot Encoding

5.4.1 Preprocessing

We ran the Isolation Forest model with one hot encoding using the following features: `hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos, pc, user, activity` where the features `pc` and `user` were one hot encoded. When applying one hot encoding on the user column, it means that each user is not represented by an id number, as they were at first, but as a column. This results in 1000 different user columns, mainly filled with zeros. Correspondingly, pc was also similarly encoded.

5.4.2 Evaluation

Since there were 1000 users and 1000 computers, it led to a matrix with 854859 rows and 2007 columns. This is too large a matrix to run the calculation using a modern consumer laptop. However, by using a large computing instance on Amazon AWS, 2007 columns are less of a problem. In 2 hours and 35 minutes the calculation finished. The instance used was AWS Sagemakers largest current generation machine learning adapted standard instance. The instance was called ml.m4.16xlarge and had 64 CPUs, 256 GB RAM and a 25 Gigabit Ethernet connection [3].

This resulted in a p value of 0.0249. This is significantly larger than without one hot encoding, which gave a p-value of 5.75×10^{-64}. In this case, a lower p-value means a better result. It is possible that the one hot encoding shifted the weighing of the features,

putting too much weight on the 2000 one-hot-encoded features resulting in a larger p-value. Considering the results in B the one-hot-encoded features pc and user are amongst the least significant which would explain the worsened results if more weight are put in these features.

5.5 Periodic Features

5.5.1 Preprocessing

As explained in Section 4.3.3 we converted all time based features from ordinal representation to periodic form. The reasoning behind this was to better mirror the time features periodicity so that the distance between two successive hours for example are always interpreted as the same. This was done by using trigonometric functions in the Python `numpy` library.

5.5.2 Evaluation

By converting the ordinal features month, weekday, and hour into periodic features, the mean p-value of the KS-test over all features, combinations was improved from 0.0274 to 0.0161, which is an improvement with a factor 1.7. The order of importance of the different combinations of features only changes slightly in a few cases when comparing the runs before and after converting the features. This could be explained by the inherent randomness of the isolation forest algorithm. The improvement of the mean p-value, on the other hand, is statistically significant with a t-score of 4.72 and a p-value of 2.47×10^{-6}.

By iterating over several limit values of the anomaly function and maximizing the MCC, we determine the best limit value. Fixing this limit to the MCC-determined value, we can calculate several parameters that can evaluate how well the model has performed, as seen in Table 5.2.

By changing the parameterization to be periodic, the accuracy (d-value) was improved by 15.65% on average when comparing all combinations of features, $15.6 = \frac{(0.602-0.52)\times 100}{0.52}$. The d value calculated using all features was increased from 0.52 using ordinal features to 0.60 using periodic features. Considering the improved p-value and the improved d value we chose to use periodic features for our model.

$$Max(MCC) = 0.04 \rightarrow limit = -0.13 \tag{5.1}$$

Table 5.2: Values calculated based on the limit determined by the best MCC value.

Best MCC	0.04
Limit	-0.139
TP	0.76
TN	0.93
FP	0.071
FN	0.24
Precision	0.0025
Recall	0.76
F1 Score	0.0050

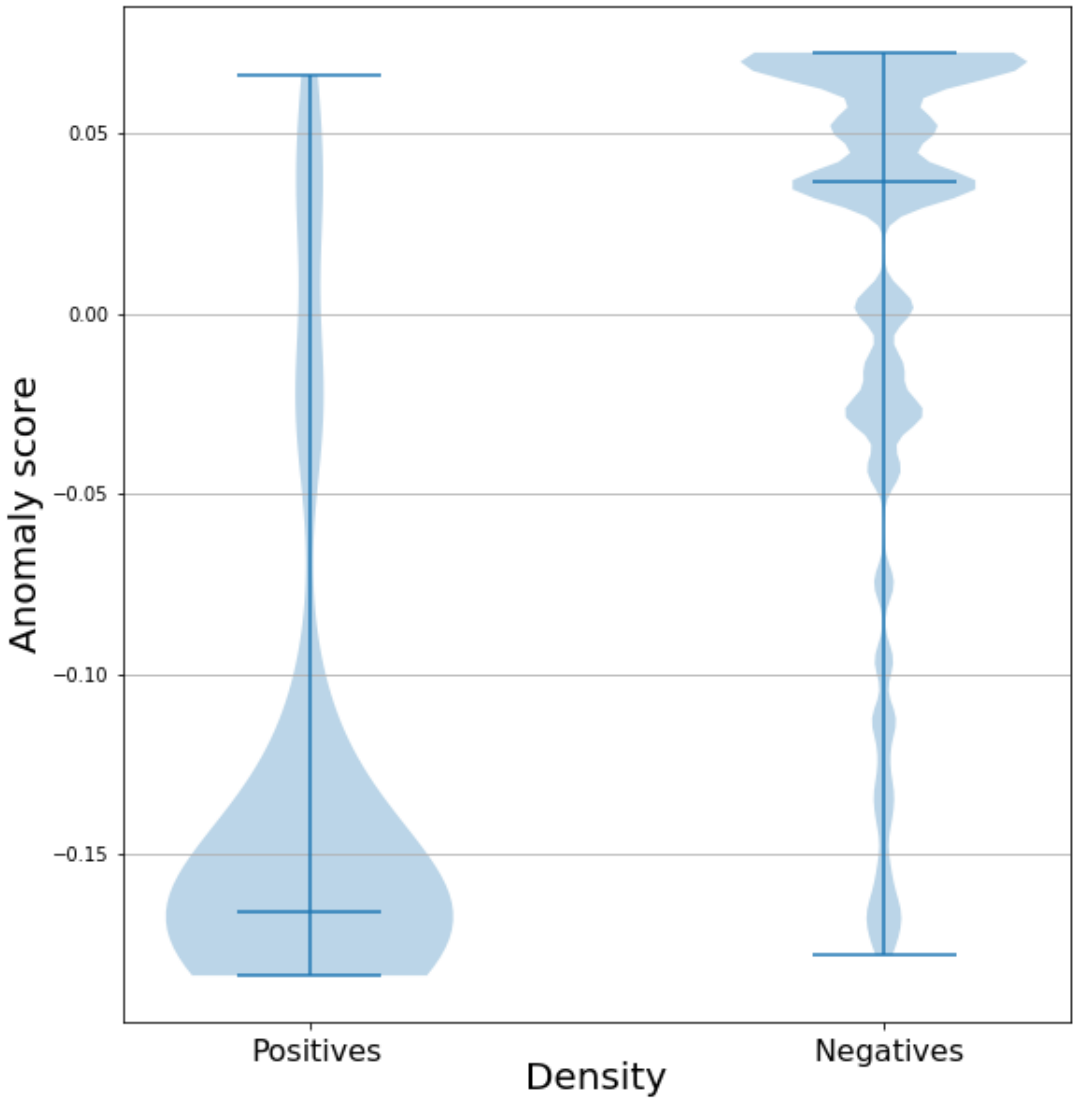

Figure 5.1: The anomaly score plotted against the density of positives and negatives respectively

As seen in Figure 5.1 the anomaly score differs substantially between the threats/positives and non-threats/negatives. The mean anomaly score is represented by a horizontal line in each distribution. The mean value for the positives were located around anomaly score -0.13 whereas the mean of the negatives is located around +0.03. This indicates that we were able to distinguish between the two classes. What were not shown in the graph was that the negatives dramatically outnumber the positives and even though the relative density of the negatives is located at a higher anomaly score compared to the positives, the absolute number of negatives located on the lower anomaly scores is still much higher compared to the positives. This, of course, is due to our unbalanced data set.

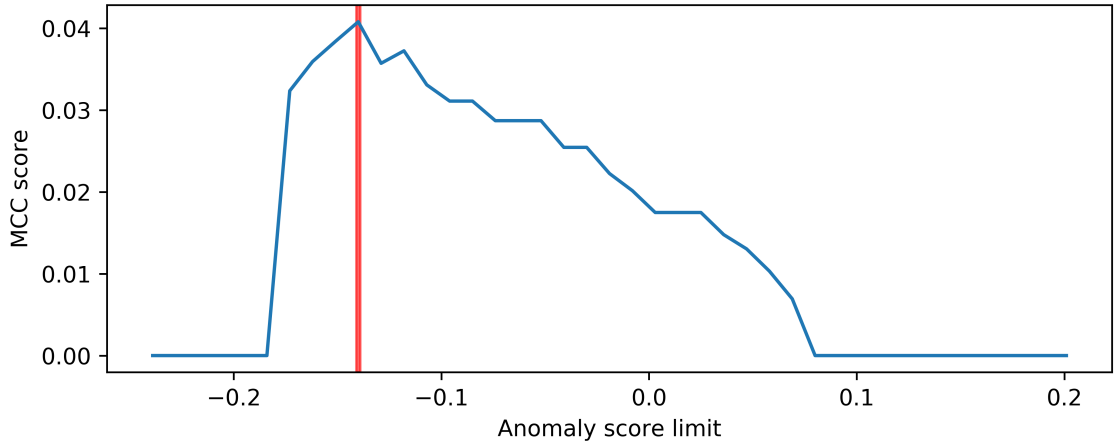

Figure 5.2: MCC score based on different anomaly score limit values for the combined features data set

Table 5.3: Confusion matrix with anomaly score limit based on MCC evaluation

Actual \ Predicted	Normal	Insider Threat
Normal	TN = 92.9%	FP = 7.06%
Insider Threat	FN = 24.2%	TP = 75.8%

Choosing the limit as -0.139 according to the maximized MCC score of 0.04, which can be seen in Figure 5.2, we receive a precision of $\frac{TP}{TP+FP} = 0.0025$. This means that by identifying 1000 actions as a potential threats we can find 2.5 actions that are real insider threats.

$$Precision = 0.0025 \tag{5.2}$$

$$IdentifiedThreats = TP + FP = 1000 \tag{5.3}$$

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{1000} = 0.0025 \rightarrow TP = 2.5 \tag{5.4}$$

The number of false positives per true positive ($TP = 1$) is:

$$Precision = \frac{TP}{TP + FP} \rightarrow 0.0025 = \frac{1}{1 + FP} \rightarrow FP = \frac{1 - 0.0025}{0.0025} = 399. \quad (5.5)$$

This means that for every one correctly identified threat, the model identifies 399 false threats. The large number of false positives that we get when optimizing the results by MCC score is due to our very imbalanced data set.

5.6 Significant features

We calculated the anomaly score for each combination of features and evaluated them using the KS-test. In Appendix B.2 we presented the results sorted by the anomaly score with the best (lowest) anomaly score on top. The top ten combinations of features are presented in Table 5.4. As it appears, the best combination of features is activity and hour.

After converting the features weekday, month and hour into periodic features we did the same evaluation to find the best combination of features. We analyzed both the sin and cos values together as a pairs, since they are related to each other. In this case also the best combination turns out to be activity together with hour_sin and hour_cos. The table is found in Appendix B.1

Table 5.4: Top 10 results using KS-Test evaluation without periodic features

Features used	KS - D Statistic	P-Value
hour, activity	0.738	6.58×10^{-96}
month, hour, activity	0.696	2.58×10^{-85}
weekday, hour, activity	0.696	2.66×10^{-85}
hour, user, activity	0.692	2.53×10^{-84}
weekday, month, hour, activity	0.689	1.57×10^{-83}
hour, pc, activity	0.687	4.08×10^{-83}
month, hour, user, activity	0.673	7.69×10^{-80}
month, hour, pc, activity	0.668	1.86×10^{-78}
weekday, hour, user, activity	0.657	4.97×10^{-76}
weekday, hour, pc, activity	0.646	1.60×10^{-73}
hour, user, pc, activity	0.639	5.81×10^{-72}

5.7 Evaluating the results

5.7.1 Anomaly score

According to the decision function in the `sklearn.ensemble.IsolationForest` library [26] the more abnormal the data is, the lower anomaly score it will have. Comparing Figure 5.3 and Figure 5.4 indicates that the Isolation Forest algorithm has successfully labeled the threat data as anomalous compared to the non-threat data. This is confirmed

when comparing the two histograms using the Kolmogorov-Smirnov test. With the null hypothesis that the two samples are drawn from the same distribution, Scipy's two-fold KS-test [11] returns a low p-value of 2.29×10^{-94} (see row 1, Table B.1), thereby rejecting the null hypothesis. The p-value is equal to the chance of the Kolmogorov-Smirnov D statistic being as large or larger than that observed, if the two samples were indeed drawn from the same distribution.

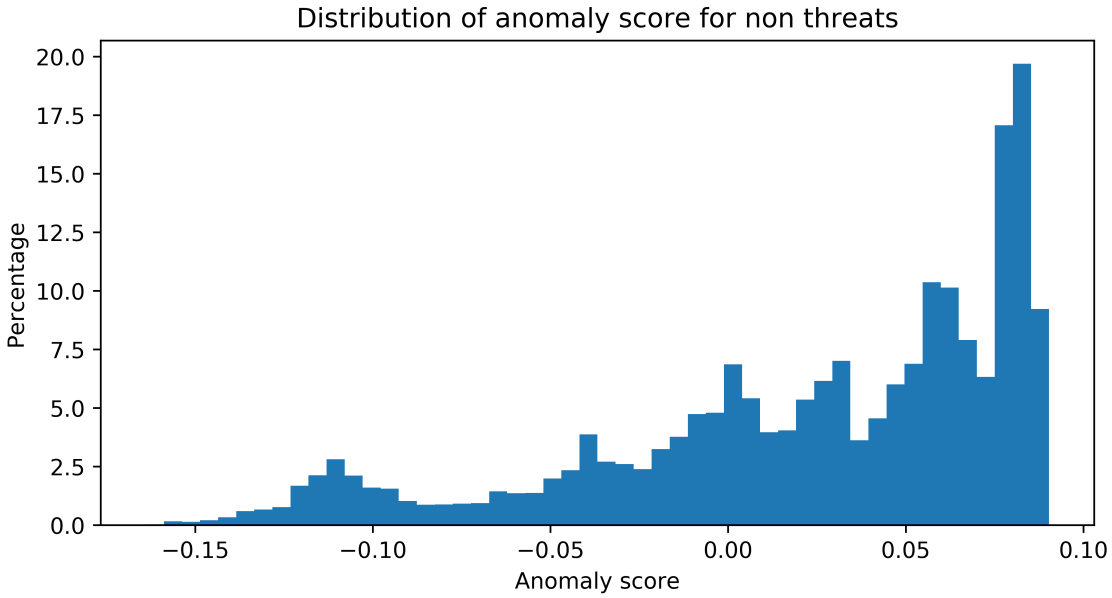

Figure 5.3: Isolation forest non threats

5.7.2 Detection rate

Through the confusion matrix for our best model, described in Section 5.3 we can see that the model was able to determine 75.8% of all the insider threats, while falsely classifying 7.06% of normal events as threats. According to the MCC test this is the best detection rate of real threats weighted against the falsely detected non-threats that our model could achieve. Whether or not this is a sufficiently good result depends on the implementation. While it might be very helpful to identify about 3/4 out of all insider threats it might also be very expensive to analyze the 7.06% false positives, especially considering how unbalanced our data set is. 75.8% out of all the insider threats in the data set corresponds to 150 threats while a rate of 7.06% false positives corresponds to 60339 events being falsely identified as threats. As indicated by the KS-tests it is obvious that the algorithm is able to isolate the threats to a degree that is statistically significant. On the other hand, as indicated by the relatively large number of false positives and by the precision value of 0.0025 there is room for improvement.

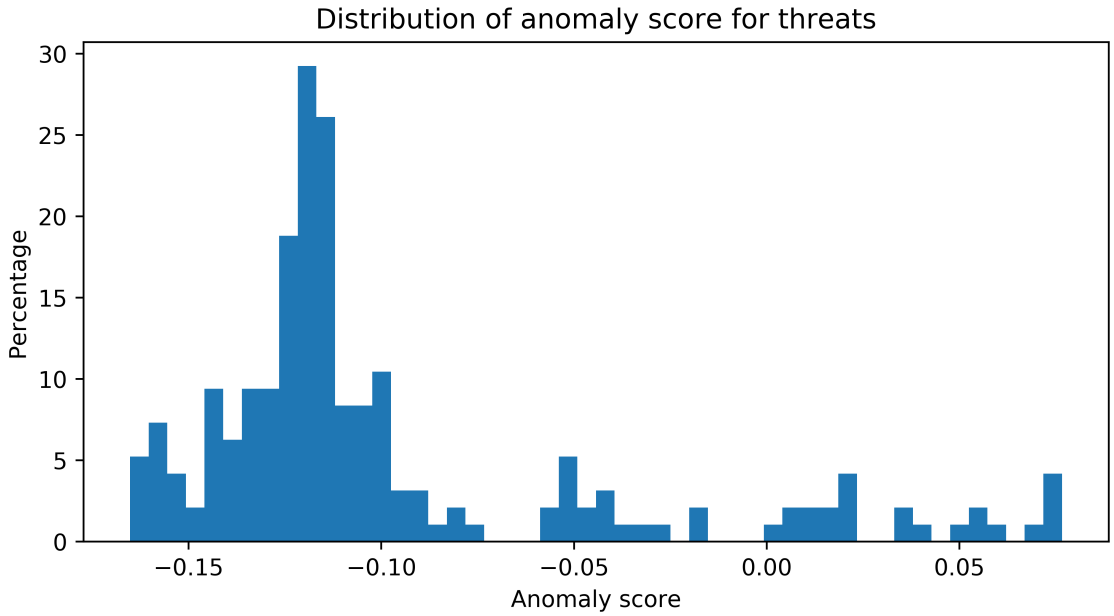

Figure 5.4: Isolation forest threats

5.8 Limitations

Only analyzing the logon data set is a significant limitation for this project. Only a certain degree of information can logically be deducted from the time a person enters and leaves work. Despite this limitations, we were able to determine 76% of the insider threats using our model. The combination of event-based detection using both logon/arrival and logoff/leaving times, as well as psychometrics, could potentially yield further interesting results. Keeping our approach of analyzing parameters attainable without constant online surveillance, adding psychometrics would be the next step in order to add additional information to solve our classification problem. Using the psychometrics data would not require any additional real time monitoring as the features in this data set does not change over time. In a real world scenario they could potentially be determined at some point during the job interview or on boarding process.

5.9 Conclusion

It is not easy to know beforehand which features are most important in order to classify the insider threat. We decided on a few ways to evaluate and quantify the results and then ran the Isolation Forest algorithm on all possible combinations of features to see which features were most important. The same method was used when evaluating different encoding methods. Using this approach we proved that the most valuable information in order to classify the insider threat lay in the time for login and logoff and that it was possible to improve the results by converting the time features into a periodic form. As the main method of evaluation we used the MCC score. By optimizing the anomaly score limit using MCC we concluded that the best performance we could get out of our algorithm

was a recall rate of 75.8%, detection rate (true positives) of 75.8% with a false positives rate of 7.06%.

Chapter 6
Conclusions

In this chapter we analyze our findings in this thesis and we discuss areas open for improvement and further research. In the analysis we present a summary of our results and an evaluation. In order to assist further research in the area, we share our insights regarding open research areas in Section 6.2.

6.1 Analysis

We were able to detect 76% of all insider threats using only the logon file. Logically, only analyzing data at this simple level, login and logoff, it should be hard to actually detect so many insider threats. Being able to detect this many insider threats, while only falsely classifying 7% of the negatives, is therefore a helpful filter. Still, given the nature of our heavily unbalanced data set and the 76% true positives and 7% false positives means that there are 399 false positives for every true positive. It will certainly require resources in a company to analyze these 400 suspected threats to detect the true insider threat. However, this cost may still be viable in certain situations since the cost of letting an insider threat take place can be much higher, and the model still filters away the bulk of the non relevant events.

An important aspect in analyzing this data is that under the scope of our thesis, we have only taken the logon data set under consideration. By simple deduction, there are and should be many insider threats that are impossible to detect using this scheme. Constructing a scheme by only taking times for log in and log out into consideration gives a limited view of the classification problem, but by using machine learning algorithms even a hard problem such as this can be solved to a certain extent.

Previously Isolation Forest is used in combination with ordinal features, but we can conclude with statistical significance that preprocessing the data using periodic features is more efficient than using ordinal features, using Isolation Forest. By converting the ordinal features into periodic features, the mean p-value of the KS-test over all features,

was improved with a factor 1.7. This is a significant improvement.

A benefit of our approach of keeping the set of features to only time of login and logout is that it may be generalizable to other methods of surveillance. For example, logging on to an office PC may be analogous to remotely logging on to the corporate network or inserting the key card at the office door. This kind of information should be relatively easily accessible for a company and compared to the features like website history or email content that is used as features in some of our related research, in Section 2, is much less intrusive when it comes to integrity of the employees.

6.2 Improvements and open research areas

We have chosen to limit the scope by only analyzing what we deemed to be the best fit machine learning algorithm for the task, Isolation Forest. The data from multiple machine learning algorithms could be compared in order to find the one best adapted for this case. Supervised and unsupervised machine learning algorithms could be compared in order to empirically validate our hypothesis that an unsupervised machine learning algorithm such as Isolation Forest is best suited for this classification problem.

In our scope we limited ourselves to analyzing the logon data set. A continuation of our research could use our approach to analyze other files in the data set provided by CERT [12]. For example, the psychometrics file. Psychometrics could be obtained at the requiting process and can be viewed as information the employee has allowed the company to use, by. would combining logon times with psychometrics give a better result, or is the extra information found in the psychometric file merely a distraction to the machine learning model, leading to poorer performance?

The feature extraction method tested in this thesis, where we combined a number of features in order to produce a new feature, timediff, did not improve the quality of the results. It is entirely possible that there are other types of feature extraction methods that could do so. At the very least there there are many others that could be evaluated.

Bibliography

[1] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN, April 2016. Online; accessed 25 May 2018.

[2] Proposal for a regulation of the european parliament and of the council concerning the respect for private life and the protection of personal data in electronic communications and repealing directive 2002/58/ec (regulation on privacy and electronic communications). http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52017PC0010&from=en, January 2017. Online; accessed 25 May 2018.

[3] Amazon sagemaker ml instance types. https://aws.amazon.com/sagemaker/pricing/instance-types/, 2018. Online; accessed 17 May 2018.

[4] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. Ensemble of binary svm classifiers based on pca and lda feature extraction for intrusion detection. pages 636–640, 2 2017.

[5] Abdulaziz Almehmadi and Khalil El-Khatib. On the possibility of insider threat prevention using intent-based access control (ibac). *IEEE SYSTEMS JOURNAL*, 2015.

[6] Amos Azaria, Ariella Richardson, Sarit Kraus, and V S. Subrahmanian. Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data. 1:135–155, 06 2014.

[7] Jason Brownlee. How much training data is required for machine learning? https://machinelearningmastery.com/

much-training-data-required-machine-learning/, July 2017. Online; accessed 27 May 2018.

[8] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[9] Davide Chicco. Ten quick tips for machine learning in computational biology. *BioData Mining*, 10, 2017.

[10] Eric Cole and Sandra Ring. *Insider threat. [Elektronisk resurs] protecting the enterprise from sabotage, spying, and theft.* Rockland, Mass. : Syngress ; [S.l.] : Distributed by O'Reilly Media, 2006., 2006.

[11] The Scipy community. scipy.stats.ks_2samp. https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.ks_2samp.html, 2014. Online; accessed 03 May 2018.

[12] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. *2013 IEEE Security and Privacy Workshops*, 2013.

[13] S. Hina and D. D. Dominic. Information security policies: Investigation of compliance in universities. pages 564–569, Aug 2016.

[14] Gareth M. James. Variance and bias for general loss functions. *Machine Learning*, 51:115–135, 2003.

[15] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen, and L. Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, April 2017.

[16] David Kaleko. Periodical features. http://blog.davidkaleko.com/feature-engineering-cyclical-features.html, 2017. Online; accessed 14 May 2018.

[17] A. Lall. Data streaming algorithms for the kolmogorov-smirnov test. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 95–104, Oct 2015.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhuo. Isolation forest. *2008 Eight IEEE Internal Conference on Data Mining*, 51:115–135, 2003.

[19] Asim Majeed, Anwar Ul Haq, Arshad Jamal, Rehan Bhana, Funke Banigo, and Said Baadel. Internet of everything (ioe) exploiting organisational inside threats: Global network of smart devices (gnsd). *2016 IEEE International Symposium on Systems Engineering (ISSE)*, 2016.

[20] Solomon Mekonnen, Keshnee Padayachee, and Million Meshesha. A privacy preserving context-aware insider threat prediction and prevention model predicated on the components of the fraud diamond. *Annual Global Online Conference on Information and Computer Technology*, 2015.

[21] Tom M Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc, 1997.

[22] pandas developers. pandas.get_dummies. https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html, 2017. Online; accessed 14 May 2018.

[23] P. Parveen, Z. R. Weger, B. Thuraisingham, K. Hamlen, and L. Khan. Supervised learning for insider threat detection using stream mining. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 1032–1039, Nov 2011.

[24] R. Rantala. Cybercrime against businesses, 2005. *Bureau of Justice Statistics Special Report*, page 6, 2008.

[25] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[26] scikit-learn developers. sklearn.ensemble.isolationforest. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, 2017. Online; accessed 03 May 2018.

[27] scikit-learn developers. sklearn.preprocessing.labelencoder. http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html, 2017. Online; accessed 14 May 2018.

[28] Li Sun, Steven Versteeg, Serdar Boztas, and Asha Rao. Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study. *CoRR*, abs/1609.06676, 2016.

[29] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[30] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *Proceedings of AI for Cyber Security Workshop at AAAI 2017*, 10 2017.

[31] David T. Wolfe and Dana R. Hermanson. The fraud diamond: Considering the four elements of fraud. *CPA Journal 74.12*, pages 38–42, 2004.

Appendices

Appendix A
Implementation Code

Our code for running the simulations is publicly available and can be found at: `https://github.com/JoakimBulow/InsiderThreatIsolationForest/`

Appendix B
Simulation results

B.1 KS-evaluation of logon data set

Table B.1: KS-evaluation of logon data set

Features used	KS - D Statistic	P-Value
activity, hour_sin, hour_cos	0.733	2.289×10^{-94}
activity, hour_sin, hour_cos, month_sin, month_cos	0.729	1.454×10^{-93}
weekday_sin, weekday_cos	0.697	1.486×10^{-85}
pc, activity, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.682	7.065×10^{-82}
activity, hour_sin, hour_cos, weekday_sin, weekday_cos	0.680	1.689×10^{-81}
pc, activity, hour_sin, hour_cos	0.678	6.086×10^{-81}
user, activity, hour_sin, hour_cos	0.674	7.122×10^{-80}
pc, activity, hour_sin, hour_cos, month_sin, month_cos	0.669	8.772×10^{-79}
user, activity, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.660	1.133×10^{-76}
user, activity, hour_sin, hour_cos, month_sin, month_cos	0.649	3.159×10^{-74}
user, activity, hour_sin, hour_cos, weekday_sin, weekday_cos	0.644	4.057×10^{-73}
user, pc, activity, hour_sin, hour_cos	0.640	4.431×10^{-72}
pc, activity, hour_sin, hour_cos, weekday_sin, weekday_cos	0.635	6.418×10^{-71}
user, pc, activity, hour_sin, hour_cos, month_sin, month_cos	0.633	1.685×10^{-70}
user, pc, activity, hour_sin, hour_cos, weekday_sin, weekday_cos	0.614	2.164×10^{-66}
user, pc, activity, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.602	5.754×10^{-64}
hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.584	4.180×10^{-60}
hour_sin, hour_cos, month_sin, month_cos	0.562	1.055×10^{-55}
hour_sin, hour_cos	0.559	3.191×10^{-55}
pc, hour_sin, hour_cos, month_sin, month_cos	0.549	3.296×10^{-53}
user, hour_sin, hour_cos	0.523	2.726×10^{-48}
pc, hour_sin, hour_cos	0.516	4.335×10^{-47}
hour_sin, hour_cos, weekday_sin, weekday_cos	0.515	6.306×10^{-47}
pc, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.510	4.962×10^{-46}
user, hour_sin, hour_cos, month_sin, month_cos	0.501	1.883×10^{-44}

Table B.1 continued from previous page

Features used	KS - D Statistic	P-Value
user, hour_sin, hour_cos, weekday_sin, weekday_cos	0.470	3.647×10^{-39}
pc, hour_sin, hour_cos, weekday_sin, weekday_cos	0.467	1.335×10^{-38}
user, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.463	6.194×10^{-38}
activity, month_sin, month_cos	0.427	2.155×10^{-32}
month_sin, month_cos	0.421	1.740×10^{-31}
user, pc, hour_sin, hour_cos	0.400	2.004×10^{-28}
user, pc, hour_sin, hour_cos, month_sin, month_cos, weekday_sin, weekday_cos	0.391	4.006×10^{-27}
user, pc, hour_sin, hour_cos, month_sin, month_cos	0.390	4.653×10^{-27}
user, pc, hour_sin, hour_cos, weekday_sin, weekday_cos	0.380	9.282×10^{-26}
month_sin, month_cos, weekday_sin, weekday_cos	0.375	4.700×10^{-25}
pc, month_sin, month_cos	0.367	5.061×10^{-24}
pc, activity, month_sin, month_cos	0.337	2.880×10^{-20}
user, activity, month_sin, month_cos	0.336	3.813×10^{-20}
activity, month_sin, month_cos, weekday_sin, weekday_cos	0.327	3.362×10^{-19}
user, month_sin, month_cos	0.313	1.536×10^{-17}
pc, activity, month_sin, month_cos, weekday_sin, weekday_cos	0.290	3.913×10^{-15}
user, pc, month_sin, month_cos	0.268	5.667×10^{-13}
pc, month_sin, month_cos, weekday_sin, weekday_cos	0.252	1.389×10^{-11}
user, pc, activity, month_sin, month_cos	0.197	3.289×10^{-07}
user	0.195	4.538×10^{-07}
user, month_sin, month_cos, weekday_sin, weekday_cos	0.189	1.081×10^{-06}
pc, weekday_sin, weekday_cos	0.186	1.693×10^{-06}
user, pc, activity, month_sin, month_cos, weekday_sin, weekday_cos	0.178	5.383×10^{-06}
user, activity, month_sin, month_cos, weekday_sin, weekday_cos	0.175	8.423×10^{-06}
user, pc, weekday_sin, weekday_cos	0.170	1.861×10^{-05}
user, weekday_sin, weekday_cos	0.163	4.203×10^{-05}

Table B.1 continued from previous page

Features used	KS - D Statistic	P-Value
user, pc, month_sin, month_cos, weekday_sin, weekday_cos	0.152	1.896×10^{-04}
user, pc, activity	0.148	2.990×10^{-04}
user, activity	0.147	3.345×10^{-04}
activity, weekday_sin, weekday_cos	0.146	3.578×10^{-04}
user, pc	0.142	6.104×10^{-04}
pc, activity	0.133	1.675×10^{-03}
pc	0.131	1.946×10^{-03}
user, pc, activity, weekday_sin, weekday_cos	0.128	2.762×10^{-03}
weekday_sin, weekday_cos	0.121	5.259×10^{-03}
user, activity, weekday_sin, weekday_cos	0.102	2.961×10^{-02}
pc, activity, weekday_sin, weekday_cos	0.096	4.836×10^{-02}
activity	0.051	6.827×10^{-01}

B.2 Without periodic features

Table B.2: KS-Test evaluation without periodic features

Features used	KS - D Statistic	P-Value
activity, hour	0.738	6.58×10^{-96}
month, hour, activity	0.696	2.58×10^{-85}
weekday, hour, activity	0.696	2.66×10^{-85}
hour, user, activity	0.692	2.53×10^{-84}
weekday, month, hour, activity	0.689	1.57×10^{-83}
hour, pc, activity	0.687	4.08×10^{-83}
month, hour, user, activity	0.673	7.69×10^{-80}
month, hour, pc, activity	0.668	1.86×10^{-78}
weekday, hour, user, activity	0.657	4.97×10^{-76}
weekday, hour, pc, activity	0.646	1.60×10^{-73}
hour, user, pc, activity	0.639	5.81×10^{-72}
weekday, month, hour, pc, activity	0.621	7.54×10^{-68}
weekday, month, hour, user, activity	0.605	1.68×10^{-64}
month, hour	0.598	5.32×10^{-63}
weekday, hour	0.596	1.62×10^{-62}
weekday, hour, user, pc, activity	0.561	1.63×10^{-55}
hour	0.559	3.19×10^{-55}
month, hour, user, pc, activity	0.548	4.96×10^{-53}
weekday, month, hour, user, pc, activity	0.521	6.173×10^{-48}
hour, user	0.508	1.414×10^{-45}
hour, pc	0.505	5.316×10^{-45}
weekday, month, hour	0.503	1.071×10^{-44}
month, hour, pc	0.493	4.788×10^{-43}
month, hour, user	0.479	1.622×10^{-40}
weekday, month, hour, pc	0.468	8.371×10^{-39}
weekday, month, hour, user	0.434	2.423×10^{-33}
weekday, hour, user	0.418	4.607×10^{-31}
weekday, hour, pc	0.416	9.301×10^{-31}
month, hour, user, pc	0.413	2.358×10^{-30}
hour, user, pc	0.371	1.452×10^{-24}
weekday, month, hour, user, pc	0.364	1.139×10^{-23}
weekday, hour, user, pc	0.355	1.513×10^{-22}
month, activity	0.348	1.197×10^{-21}
month	0.348	1.328×10^{-21}
weekday, month	0.235	4.379×10^{-10}
weekday, user	0.210	4.035×10^{-08}
month, pc	0.206	7.532×10^{-08}
month, pc, activity	0.199	2.168×10^{-07}
pc, activity	0.192	6.987×10^{-07}

Table B.2 continued from previous page

Features used	KS - D Statistic	P-Value
weekday, user, pc	0.188	1.214×10^{-06}
weekday	0.186	1.839×10^{-06}
weekday, month, activity	0.185	2.104×10^{-06}
user	0.173	1.166×10^{-05}
weekday, month, pc	0.163	4.526×10^{-05}
month, user	0.162	4.784×10^{-05}
user, pc	0.162	4.920×10^{-05}
weekday, user, pc, activity	0.160	6.967×10^{-05}
user, pc, activity	0.155	1.172×10^{-05}
weekday, pc	0.141	6.681×10^{-04}
weekday, user, activity	0.141	6.927×10^{-04}
user, activity	0.140	7.565×10^{-04}
pc	0.138	9.287×10^{-04}
month, user, pc	0.132	1.754×10^{-03}
weekday, activity	0.121	5.259×10^{-03}
month, user, activity	0.113	1.134×10^{-02}
weekday, month, pc, activity	0.111	1.342×10^{-02}
weekday, month, user, activity	0.102	3.117×10^{-02}
weekday, pc, activity	0.088	8.94×10^{-02}
month, user, pc, activity	0.084	1.13×10^{-01}
weekday, month, user	0.083	1.24×10^{-01}
weekday, month, user, pc, activity	0.072	2.48×10^{-01}
weekday, month, user, pc	0.063	3.97×10^{-01}
activity	0.051	6.82×10^{-01}

Appendix C
Information from data set

This is the contents of the readme.txt file in the data set.

```
Release 4, Dataset 2 Notes

Major Changes
* Content is integrated with the graph structure.
* A user's topics of interest can drift over time.
* Email now includes CC/BCC.
* Email table now includes user ID and PC.
* Users can have one or more non-work email addresses.
* A latent job satisfaction variable was added.
It might make sense for us to specify exactly
how this affects observable variables, so let us know if that information is desired.
* An additional red team scenario was added.
(All previous red team scenarios also occur in the dataset.)
* This is a "dense needles" dataset.
There is an unrealistically high amount of red team data interspersed.


license.txt
* ExactData license information


logon.csv
* Fields: id, date, user, pc, activity (Logon/Logoff)
* Weekends and statutory holidays (but not personal vacations)
are included as days when fewer people work.
* No user may log onto a machine where another user is already logged on,
unless the first user has locked the screen.
* Logoff requires preceding logon
* A small number of daily logons are intentionally not recorded to simulate dirty data.
* Some logons occur after-hours
  - After-hours logins and after-hours thumb drive usage are intended to be significant.
* Logons precede other PC activity
* Screen unlocks are recorded as logons. Screen locks are not recorded.
* Any particular user's average habits persist day-to-day
  - Start time (+ a small amount of variance)
  - Length of work day (+ a small amount of variance)
  - After-hours work: some users will logon after-hours, most will not
* Some employees leave the organization:
```

```
no new logon activity from the default start time on the day of termination
* 1k users, each with an assigned PC
* 100 shared machines used by some of the users in addition to their assigned PC.
These are shared in the sense of a computer lab, not in the sense of a
Unix server or Windows Terminal Server.
* Systems administrators with global access privileges
are identified by job role "ITAdmin".
* Some users log into another user's dedicated machine from time to time.


device.csv
* Fields: id, date, user, pc, activity (connect/disconnect)
* Some users use a thumb drive
* Some connect events may be missing disconnect events,
because users can power down machine before removing drive
* Users are assigned a normal/average number of thumb drive uses per day.
Deviations from a user's normal usage can be considered significant.


http.csv
* Fields: id, date, user, pc, url, content
* Has modular/community structure, but is not correlated with social/email graph.
* Domain names have been expanded to full URLs with paths.
* Words in the URL are usually related to the topic of the web page.
* Content consists of a space-separated list of content keywords.
* Each web page can contain multiple topics.
* WARNING: Most of the domain names are randomly generated,
so some may point to malicious websites.
Please exercise caution if visiting any of them.


email.csv
* Fields: id, date, user, pc, to, cc, bcc, from, size, attachment_count, content
* Driven by underlying friendship and organizational graphs.
* Role (from LDAP) drives the amount of email a user sends per day.
* The vast majority of edges (sender/recipient pairs)
are exist because the two users are friends.
* A small number of edges are introduced as noise.
A small percentage of the time, a user will email someone randomly.
* Emails can have multiple recipients
* Emails can have a mix of employees and non-employees in dist list
* Non employees use a non-DTAA email addresses;
employees use a DTAA email address
* Terminated employees remain in the population,
and thus are eligible to be contacted as non-employees
* A friendship graph edge is not implied between
the multiple recipients of an email.
* Unlike the previous release, we do not believe
the observed email graph follows graph power laws
because the power-law-conforming friendship graph is
overwhelmed by the organizational graph.
* Email size and attachment count are not correlated with each other.
* Email size refers to the number of bytes in the message, not including attachments.
* Content consists of a space-separated list of content keywords.
* "Content" does not specifically refer to the subject or body.
We have not made that distinction.
* Each message can contain multiple topics.
* Message topics are chosen based on both sender and recipient topic affinities.


file.csv
Fields: id, date, user, pc, filename, content
* Each entry represents a file copy to a removable media device.
* Content consists of a hexadecimal encoded file header
followed by a space-separated list of content keywords
* Each file can contain multiple topics.
* File header correlates with filename extension.
* The file header is the same for all MS Office file types.
* Each user has a normal number of file copies per day.
Deviation from normal can be considered a significant indicator.
```

```
psychometric.csv
* Fields: employee_name, user_id, O, C, E, A, N
* Big 5 psychometric score
* See http://en.wikipedia.org/wiki/Big_Five_personality_traits
for the definitions of O, C, E, A, N ("Big 5").
* Extroversion score drives the number of connections
a user has in the friendship graph.
* Conscientiousness score drives late work arrivals.
* This information would be latent in a real deployment,
but is offered here in case it is helpful.
* A latent job satisfaction variable drives some behaviors.

Malicious actors
* This data contains two instances of insider threats.
* Data dimensions that are fair game for anomaly detection
(not all are used in red team scenarios)
  - In general, radical changes in behavior
  - Unusual logon times (for that user)
  - Unusual logins to another user's dedicated machine
  (for users that don't do this normally)
  - Device usage for users who aren't normally device users,
  or increased device usage for those that are.
  - Radical increases in the amount of device usage by a user
  - Employee termination
  (as an indicator, but not anomaly detection per se)
  - Number of emails sent / day
  - Change in web browsing habits
  (visits to unusual websites are interesting, but also common)
  - Radical change in social graph behavior
  (unexpected email recipients, perhaps)
  - Topics of web sites visited, emails, and files copied.
* We can reveal as much as you would like about the red team scenarios.
* This is a "dense needles" dataset.
There is an unrealistically high amount of red team data interspersed.

Errata:
* Field Ids are unique within a csv file (logon.csv, device.csv)
but may not be globally unique.
```