How does enhancing user activity features in Machine Learning algorithms offer a pre-emptive counter measure to detecting corporate insider threats?

MSc Internship
Cyber Security

Brian Redmond
Student ID: 17137250

School of Computing
National College of Ireland

Supervisor: Mr Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Brian Redmond		
Student ID:	17137250		
Programme:	Cyber Security	**Year:**	2018
Module:	MSc Internship		
Supervisor:	Mr Ross Spelman		
Submission Due Date:	17/8/2020		
Project Title:	How does enhancing user activity features in Machine Learning algorithms offer a pre-emptive counter measure to detecting corporate insider threats?		
Word Count:	**8818**	**Page Count**	**29**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: [illegible]

Date: 14/8/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

How does enhancing user activity features in Machine Learning algorithms offer a pre-emptive counter measure to detecting corporate insider threats?

Brian Redmond
17137250

Abstract

Despite extensive research within the security community, the objective of finding scalable and accurate solutions to the problem posed by corporate insider threats has never been greater. Continuous and growing challenges exist for corporation and government entities to protect their environment from malicious and inadvertent exposure of sensitive data from within their own organisation. In this research paper, we examine the use of Machine Learning Algorithms, building advanced feature definition, analysing and classifying data patterns in user activity to put forward a conceptual model to mitigate the risk of data loss due to insider threats. This work extends existing research in the area of using anomaly detection and classification learning algorithms. We evaluate supervised learning, crossing threat features for classification and building data profiles for employees. We use existing industry CERT datasets on insider threats along with synthetic injected data into the Apache Spark Machine Learning library. Our objective is to analyse through developing advanced feature extractions that the accuracy levels support the detection of suspicious activity. This will be evaluated by building an application to analyse streaming data and design a feature classifier merging real time activity with historic patterns for an individual employee. The result of the research puts forward a method for developing activity feature extraction, shows an average level of accuracy level of 99.984%.

1 Introduction

An insider, generally referred to as an employee or contractor can be defined as an individual or a process directed by an individual who can adversely impact the confidentially, integrity or availability of an organisation's information store. The threat carried out by the individual can be malicious involving theft, fraud, espionage or unintentionally releasing sensitive information.

Current research has focused on a number of areas, filtering or sentiment analysis which interprets user communication and based on textual content and will attempt to highlight a potential threat. Anomaly based algorithms is a new field of research using machine learning technology in an attempt to detect deviations from the baseline. Once a breakout occurs this is classified as a threat.

With exponential data growth in the last number of years, this has presented a new challenge for security professionals and organisations. In 2017 the SANS institute outlined in their industry survey that threats from within an organisation are now considered higher than external threats, with 63% of organisations outlining data compromise internally as their number one security concern (Cole, E. SANS Institute; 2017).

The impact of losing confidential corporate sensitive client data can have a major impact on an organisation's cost, reputation and in some cases the ability to continue operations. It is estimated that the average cost in 2018 was $3.86 million per breach (Ponemon Institute. and IBM; 2018). High profile cases such as Sage Accounting in 2016 where 300 business clients had sensitive information released by a single disgruntled employee resulted in a share price drop of 4.3%.[1] In 2016 a Google engineer had the capacity of downloading 14,000 files before leaving the organisation and subsequently sentenced to 18 months for trade secret theft.[2] Even despite the number of high profile cases, the extent of the problem can be difficult to gauge as organisations fail to disclose or delay the extent of a data breach.

This poses a challenge. Traditional methods of security controls to reduce the attack surface are employed by organisations. Firewalls, Intrusion Detection, authentication and authorisation tools are all effective in detecting external threats, Jang-Jaccard and Nepal (2012) however less effective in combating insider threats. The new challenge which grows in complexity is from inside activity where users, in some cases with elevated privileges, understand security procedures and have access to and knowledge of applications. A recent PwC Audit committee [3] outlined that 90% of insiders displayed no concerning characteristics prior to the attack, with 80% committed during the working hours. This makes detection in a large data set difficult to isolate with more complex solutions required.

The motivation behind this research is driven from the various challenges outlined. According to the SANS institute insider threat survey in 2016, 62% of organisations surveyed have never experienced an insider attack however 38% will admit to have ineffective prevention measures. Cole, E. SANS Institute; (2017). This shows that a current solution still has to evolve to meet the requirement.

Research has advanced in the area of behavioral detection, but has still to address a number of short falls. The challenge in this area of research has been the high level of false positives, the definition of normal activity and the acquisition of training data. For example, with the addition of new applications on to the network, training data to isolate an anomaly is difficult to define. Organisations become as insecure as their weakest point, which reduces the effectiveness of the overall model. In addition, key success factors are to understand the insider motivation and behavior. Current research in psychological behavior has attempted to address shortcomings in traditional detection methods by using personality traits and applying models which suggest individual patterns of such usage as browsing history can provide evidence of a potential threat. Advances have been made, but still fall short on practical implementation and integration with other factors.

This paper presents a new method for pre-processing data into cross features for presentation to a machine learning algorithm. The objective is to consider that by crossing user activities as features and producing a general threat model if there is a potential for a higher level of success in detecting a threat.

[1] https://www.reuters.com/article/uk-cyber-sage/sage-customers-exposed-to-data-breaches-of-their-own-making-idUKKCN10T18Q
[2] https://www.bbc.com/news/world-us-canada-53659805
[3] https://www.pwc.co.uk/audit-assurance/assets/pdf/insider-threat-for-google.pdf

The background of this research is to analyse the requirement within the industry focusing on the problem of threats coming from individuals, employees, contractors within the organisation. The diversity of the behaviour patterns outline by Bulling et al.; (2008) defines the scale of the problem. The complexity of enterprise applications, the volume of information stores and the sheer number of use cases, point towards a scalable technology. Machine learning offers a solution to the scale and taking knowledge of workflows of the specific insider threat and applying a mathematical model. The model based on the dataset supplied defines features and training algorithms provides a statistical prediction of the result. This research will develop a feature profile based on a user action and predict the potential of that action being a threat. Apache Spark is the Machine Learning engine chosen for this research. Apache Spark is a cluster computing framework giving the ability to execute processes across multiple nodes in a cluster and combine the results into one result set. When rated against other engines Apache Spark has shown increase scalability with faster runtimes. Garcia-Gil et al.; (2017)

We analyse the profile of user activity by taking each action, logon, email, URL, file access and device connect and ask the question how we can develop a generic method with high probability of success to build up a feature set for presentation to the Machine Learning algorithm. We analyse if there is sufficient quantitative data to develop a hypothesis that providing machine learning algorithms with advanced features is an effective measure at predicting and pre-empting an insider threat.

We develop a series of data models stored in structured schemas loaded into a database, build a probability extraction APIs, present this to the ML training and testing module and develop a framework for real-time simulation testing. By enhancing the probability features we conducted a series of experiments aimed at testing if the model can effectively predict the probability of an internal attack.

This paper focuses on using CERT industry data, with additional synthetically injected threats into the dataset. For the test Apache Spark is deployed using Random Forest algorithm.

Section II. Discusses all the related work in this field of research, defines what the current status and limitations are within this research field.
Section III. Outlines the method used for data gathering and preparation and defines the framework for testing.
Section IV. Outlines our approach to designing the model.
Section V. Provides implementation logic.
Section VI. Evaluates the outcome of the testing.

2 Related Work

As far back as 2005, Theoharidou et al.; (2005), defined the term ‘insider Threat’ as a misuse of privileges by individuals who have been granted rights to system information. CERT extends the definition in a number of ways (Dawn et al.; 2012), specifically extending the scope, identifying the organisation, the asset, unintentional and business partner.

When we consider insider threats, the key problem with this definition is as technological innovation continually increases, the definition does not go far enough. The insider threat platform is expanding. Services internally developed, constantly communicating with each other through for example APIs which have now to be added into the definition.

Historically definitions do not address motivation of the insider and elements of motivation are important in threat definition terms as the threat may grow over time. Yang [6], in considering threat detection using Psychological profiling makes the point that the threat evolves in individuals over time.

2.1 Psychological Detection Approach

As part of the overall approach to an insider threat, it is important to analyse Motivation and Psychological factors of the threat actor. This is important to our research if this is a factor which could influence or be taken into account within our dataset. To provide comprehensive research leading to a potential solution, we must consider inclusion or exclusion. To this end we study and take a view from Yang et al.; (2018) Potential Malicious Insiders Detection Based on a Comprehensive Security Psychological Model. Yang highlights, that although traditional detection methods deter inside attackers using both anomaly detection and policy violation checks that research has failed to address the authenticated insider under normal behaviour. Yang argues that comprehensive analysis of the combination of both Technical monitoring and profiling is required. This argument has also been put forward in Maasberg et al.; (2015) and Sood et al.; (2017). Sood et al.; (2017) would further argue that psychological model derived from Big-5 and Dark Triad personality traits provide evidence that insider threats can be isolated in advance of the action.

However scholars are divided and Youli and Chao (2014) highlights that although significant progress has been made that problems particularly in the area of situational factors still exist and further research is required. This is an important within our area of research as one of the goals of research is to investigate the potential for the reduction of false positives. If situation factors exist, the element of false positives will increase. For example, if an employee has put forward strong resistance to a certain organisation's strategy in the area of environmental issues. Should this be a factor in relation to a potential insider threat?

We conclude that although there is merit in the concept of isolating a specific target group as a counter measure that this approach falls short to detect focused sophisticated or unintentional actors and as such could provide only a partial solution. With our model of defining algorithm for pattern isolation we should be able to include this group in the models detection. One question that needs to be asked is how this category of data is acquired and maintained plus what the restrictions in relation to obtaining this data are.

2.2 Sentiment Detection

One of the issues associated with the anomaly insider detection approach is the high amount of false positives returned. Sentiment analysis approach is the prediction and detection of malicious insider's motivation based on sentiment profiling while browsing webpages and within the content of their emails.

Browsing history and textual content of emails are scanned and matched against a serious of predefined words or phrases to build up a sentiment matrix. Jiang et al.; (2018) puts forward a comprehensive sentiment analysis and malicious URL detection algorithm to address potential insider threat detection. Logs are interrogated to search out phrases such as Fraud; Bribe etc. 400 such words and phrases are compared for the sentiment to build over a period of time. Legg et al.; (2017) extends this concept to include additional factors such as obtaining data from physical activity, logging at entry into rooms within the building. With a more detailed data set the user based profile is enhanced. The data set is then built into a tree structure and the core algorithm then compares one user profile against the other in an attempt to isolate anomalies.

Three key serious weaknesses exist with this approach.

1. Sentiment builds up over time. Profile of activity is built and intelligence makes a decision that it could be a potential insider threat. However when the profiling is complete the malicious attack could have already taken place. This is a general theme in relation to a number of approaches we have outlined. Timing is a key factor in evaluating a potential insider threat. Sentiment and Psychological, all build an excellent image of a potential threat and the false positives once the profile is complete is low, however there is no reference to the time this takes and to the fact that the attack could have already taken place.

2. Unsophisticated attackers could conceivably be caught out by the sentiment trap however it is less likely that sophisticated attackers would be. These consist of a group of highly motivated, malicious attackers aware of the surrounding environment and are prepared for such filtering.

3. Organisations specifically have policy and training towards sentiment content which is used within these approaches. The use of proxy servers to restrict browser activity, email filtering will restrict the effectiveness of this approach. We conclude that sentiment analysis has a place in insider threat detection but only covers one element. Sentiment would yield results while an inside attack is being planned. Sentiment could be misleading for example when an organisation changes policy on working conditions, this could lead to indifferent sentiment to be found in employees emails, instant messages, and web browsing.

2.3 Anomaly Detection using Supervised and Unsupervised learning Algorithms

The application of Machine Learning (ML) and Artificial Intelligence has become a popular area of research in the field of Cyber insider threats in the last number of years. Machine Learning provides powerful prediction capabilities when supplied with big datasets. The processing capability has also been increased with the scalability made available from cloud environments. There are primarily three (ML) algorithm types, unsupervised aims to learn the structure of data without been explicitly provided and it becomes useful, as it automatically identifies the structure from the dataset. Supervised learning is where data has been classified already and requires pre-analysis. Reinforcement is a hybrid of both types. Security controls and insider threat detection have seen a lot of research in the area of machine learning to isolate anomaly detection, however the accuracy rate still remains low. When we analyse insider threat detection there are a number of factors which must be taken in consideration. Meng et al.; (2018) puts forward an approach to tackle two key challenges, redundancy and non-linear relationships. Meng puts forward a technique (PCA) Principal Component Analysis, which is used to highlight or define distinct patterns in a dataset. It measures the distance of each data item from the normal distribution. The process is repeated until deviated data items are normalised and the result set is a series of unique deviations or anomalies. This forms the underlying concept of anomaly detection in insider threats. The advantage of this method is that the data items or variables can be un-correlated or unsorted. In practical terms, each activity by a user within an application becomes a data item. It allows the model to accept dispersed data from for example system logs. Machine Learning is then employed to perform the algorithmic mathematical underlying analysis. Meng puts forward the usage of Deep Learning as a solution for performs this.

The concept is that Deep Learning is appropriate to model non-linear relationships by extracting unrelated data into multiple layers. The layer represents a data item and this data item in practical terms could be a user running a report from an application. Meng et al.; (2018) then considers the challenge of performance in big datasets. In order to satisfy the volume of data required to meet the needs of the Machine learning model the latest CERT data v6.2 is used. This question of performance was equally put forward in research by Tuor et al.; (2017) and both papers suggested that Long Short term memory Recurrent Neural Network (LSTM-RNN) which had achieved considerable results in the area of natural speech recognition would be employed. (LSTM-RNN) as the name suggests, Recurrent, maintains persistence. For example like human thoughts are built on previous thoughts. A human's understanding of their current thought is realised from many other previous thoughts. (LSTM-RNN) is built on the same logic. Neural networks which maintain a network of individual potentially unique nodes i.e. unrelated activity, actually falls short, however a recurrent neural network which constantly loops information from the previous node into the current node is the fundamental underlying concept of (LSTM-RNN) . Meng et al.; (2018) and Tuor et al.; (2017) developed this approach in their papers to address performance and learning. Both papers introduce user dataset activity from the CERT dataset as attribute classifiers to feed the network.

Meng et al.; (2018) sets out in their paper an accuracy goal at 93.85% for success. Using a number of methods 89.54 % is achieved with the Isolation Forest method (IF). The (IF) method commonly used in fraud detection is one of the newer formulations and isolates observations in the data. From the observation defines randomly split values and takes min/max values to compare against other observations. Meng et al.; (2018) conclusion is that this approach achieves higher performance against previous strategies. However Tuor et al.; (2017) outlines an approach from the same CERT dataset which obtains a 95.53% accuracy rate which is different from Meng et al.; (2018) Isolation Forest and Support Vector Machine (SVM) and (PCA) methods. Tuor notes that because of the variance in insider threat behaviour it required the element of streaming of data into the network. Tuor give us an example of a user copying an unusually large file during a certain period of time during the day, however it doesn't address the classification that this could be a legitimate transaction.

Tuor et al.; (2017) has outlined the best approach to date. However states in the conclusion that the architecture employed is only an indicator or potential of a data breach and however although it outperforms previous models (IF, SVM and PCA) which is what the research aimed to achieve, it fails to apply any understanding of the action i.e. the large file movement could be legitimate and offers no potential solution to enhancing this approach with further data. Tuor also fails to fully define what the previous performance is that the research compares against. It must be noted that Tuor states in the conclusion that an extension of the research would be to widen the dataset and provide more granularity of the data.

Additional research was performed within this area by Le et al.; (2018) who extended the supervised and unsupervised learning algorithms to use Self Optimising Maps (SOM), Hidden Markov Models (HMM) and Decision Trees (DT). Both HHM and SOM are different in their approach in that they attempt to provide some probability to the next node. These models work well in a supervised state where structure is known and if the baseline is breached the action is reported for a decision. We see this method used in for example banking transactions where a suspicious transaction is referred and decision is made on the release or rejection of the transaction. Work performed by Le et al.; (2018) is comprehensive and conclusive that SOM gives better performance. Equally as with Tuor and Meng, Le specifically notes that streaming of data for real time detection is the next step in further research.

2.4 The Challenge Ahead

In 2017 Zaytsev et al.; (2017) produced a critical analysis research paper on the area of insider threats. The aim of this paper is to analyse current thinking and the approach taken in the research field of insider threats, develop a taxonomy for insiders, attacks and potential countermeasures, investigate current forecasting models and define the concept of early detection and the role that it plays.
A comprehensive study of insider behaviour, counter measures employed and modelling outcomes were investigated. The paper concludes that despite the amount of research in defining the problem and counter measures employed the prediction of Insider behaviour using modelling has not yet been developed.

Hall et al.; (2018) begins to research general purpose models for prediction and we consider if the model can predict the threat in real time which application can make a decision to reject the action. We compare a single generic model with multiple features against separate model to detect each separate threat scenario.

In the case of this research the data we will deal with is user activity and we will make this activity as granular as feasible. Zhang et al,; (2017) outlines the potential capacity. In experimentation Zhang pushes the randomisation of the data to levels which allow the algorithm to continue readdress the model without fail.

3 Methodology

Our approach to developing the environment is broken down into 4 stages:

1. Data accumulation and pre-processing.
2. Transform and process the dataset.
3. Execute a series of tests.
4. Analyse the results.

Figure 3.1 shows the high level overview of research stages with a breakdown of the functional requirements.

Design the data model	SQL loaders to create the relational database model. Cross Feature Extraction Loaders Build up threat data

Design and build the threat application	Feature Builder for analysing data blocks and building features sets Probability builder API to apply probability to features Machine Learning prediction module Synthetic Threat injection module Scheduler for execution of test data sets
Design the series of lab test experiments	Scheduler to execute test scripts Logging module to maintain results details
Result Analysis	Evaluator to produce analysis of output

Figure 3.1: High level Overview

Figure 4.1 outlines the approach to the physical design stages of the application required to build the insider threat environment. The first stage was to move the physical data from the CERT dataset to a relational database model. A series of data loaders was developed, building up a single view of the data. Synthetic threat records were injected into the dataset through threat loaders.
The feature API module was built with 2 layers.

1. Individual layer creating a feature for an individual activity, logon, email, device connect.

2. The Cross feature extraction layer takes (1) in real-time and build up a profile of a user's activity over a defined period of time. This defined period or interval can set the algorithm to gather data on that user over days, weeks, months. This was developed within the data preparation API to query the database for multiple activity types and calculate the probability of each activity while building features.

A probability builder module was developed to apply a probability score to features. A feature is defined as a measure applied to the characteristic of an action. Our approach to the application of feature measure is in 3 different ways.

1. Z-scores to show the distance of the individual action from the mean.
2. URL weighting where specific scores are applied to URL content.
3. Cross feature extraction where a hybrid of Z-scores and weighting is applied to a series of features.

Test experiments required the creation of a scheduler module in order to simulate durational workflow production. The scheduler can execute a series of tests for a given duration in time. We enter from to variables into the scheduler which executes the model each hour for that duration. This enables us to view our output data before and after a threat transaction.

The application to build the test and simulation environment was built using a JAVA 8 backend with Apache Spark 2.4 machine learning library (ml). MySQL database was used to store the relational data, deployed using XMAPP v.7.2. The application was deployed on a Windows 2016 VM. IntelliJ dev. was the chosen development IDE due to it's rapid development framework capability and ease of plugins. Apache Spark was the data processing framework chosen. The ml library was employed over mllib to avail of dataframes

rather than Resilient Distributed Datasets (RDDs) , which gives additional data processing speed. A series of SQL loaders scripts was developed to load the initial CERT dataset into a MySQL data. CERT dataset has 2 versions r6.2 and r4.2. r4.2 has 1000 employees spread over 17 months with 30 malicious employees and 32.7 million activities.
Results are evaluated using industry standard metrics, accuracy and confusion matrix output for machine learning algorithms. We use the confusion matrix to describe the performance of our model, comparing the predictive versus actual labels and how many did the classifier get correct.

Testing strategy was expanded in 3 areas:

1. Expand the feature set to the nth degree and analyse if there is a higher level of detection
2. Real-time slice. A scheduler module has been developed to move through the dataset and continuously run the application for a parameterised time, for example each hour.
3. Introduce threats at discrete time slices.

The approach we took to building features for user profile takes a single action, login, files access, URL access and creates a feature classification for that action and pulls in data for a prior defined interval period. This period was set as a parameter for testing our model, (n-x) prior days / hours / weeks. The approach has proven strong in detecting the pattern in a potential threat. However we recognise a limitation in this approach to developing the feature extraction. Further discussed in the conclusion this fixed interval needs to evolve into a more advanced approach.

This approach builds up the various stages to allow the application to answer our research question. The environment is setup to test our cross feature algorithm at a given instance in time and vary our algorithm's configuration to isolate the threat in real time.

4 Design Specification

As outlined in our methodology we have built an end to end simulation insider threat environment with lab testing capabilities. Design specification consists of data processing, system architecture and test design.

The data repository is built from 3 stages:

1. Raw data processed from the CERT dataset with synthetic injected transactions.
2. Feature definition from the dataset.
3. Probability rule set application to features.

Figure 4.1 illustrates the dataflow and transformation of information.

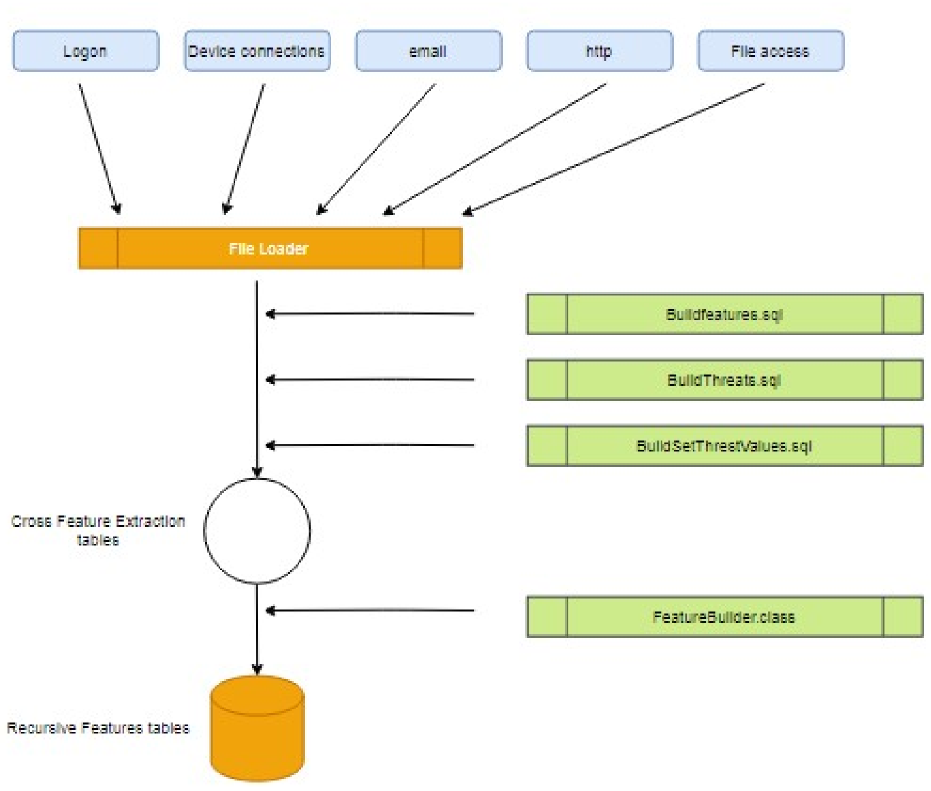
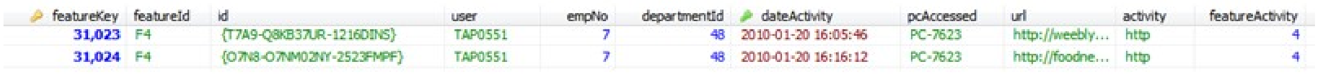


Figure 4.1 Data warehouse dataflow

SQL script loaders take source data from a series of csv files and load them directly into the data warehouse tables. Additional lookup tables are derived, user details, departments, activity. Once loading is complete, a data integrity script verifies the insert of the data. Initial features fields are built with data pre-probability calculation. Figure 4.2 illustrates initial non feature table fields.



featureKey	featureId	id	user	empNo	departmentId	dateActivity	pcAccessed	url	activity	featureActivity
31,023	F4	{T7A9-Q8KB37UR-1216DINS}	TAP0551	7	48	2010-01-20 16:05:46	PC-7623	http://weebly...	http	4
31,024	F4	{O7N8-O7NM02NY-2523FMPF}	TAP0551	7	48	2010-01-20 16:16:12	PC-7623	http://foodne...	http	4

Figure 4.3 Non feature fields

The data warehouse consists of 2 dataset types with the following records:

CERT Dataset r4.2	LDAP 16,743	email 2,629,979	File access 445,581	Device access 405,380	logon 854,859	Http 23,434,423
Synthetic threats	logon	email Unlimited	Http			

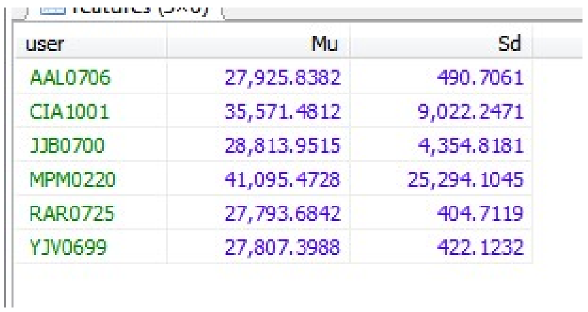
Threat data is generated to allow the testing module to analyse detection of certain conditions not available in the cert data. Using the same user base and access times as CERT data we extend additional transactions with reporting access, client access. This allows the test engine to analyse additional feature classification.

Data for all individual activities with interval feature probabilities are stored in a single table for processing within the threat monitor machine learning module.

The FeatureBuilder java class calculates and applies probability or rating factors to all features. We measure features in terms of probability using Z-scores. Z-scores allow us to apply a measure of how far an individual action is away from the mean. Take one example, finding the Z-score for a logon time. The question is if a single logon is normal / abnormal i.e. has a user logged in out of business hours that would not normally logon at this time. Abnormal would suggest a threat however in conjunction with other features, could this suggest a higher potential threat? We use this logon probability as an illustration of the way calculation takes place across all the features in our model. We calculate **μ** (mean) and **σ** (standard deviation) for a given period then we apply the action value to a Z score formula

Z-score = $(X - \mu) / \sigma$

This answers the question for each logon of how many standard deviations from the population it is. For the logon time we convert time to seconds using TIME_TO_SEC, i.e. select TIME_TO_SEC (dateActivity). Then for each user, we calculate the mean and standard deviation for the entire dataset.
select user, AVG(TIME_TO_SEC(`dateActivity`)) as Mu , STDDEV(TIME_TO_SEC(`dateActivity`)) as Sd from features where activity = 'Logon' and featureId = 'F1' group by user;



user	Mu	Sd
AAL0706	27,925.8382	490.7061
CIA1001	35,571.4812	9,022.2471
JJB0700	28,813.9515	4,354.8181
MPM0220	41,095.4728	25,294.1045
RAR0725	27,793.6842	404.7119
YJV0699	27,807.3988	422.1232

The same logic is then applied to each of the features. In different cases a logical time slice is applied. For example when we analyse email activity, we want to ascertain in conjunction with other features are email attachments of an excessive size being sent in a given period

In addition we apply 2 different types of measure:

- Normal values, the time of which a user accesses the application.
  Monday to Friday = 0 Saturday = 1, Sunday = 2
- Value weighting. If the URL accessed for example contains 'WikiLeaks' or 'KeyLogger' we weight high - 10 else 0.

The key to our research is the approach we take to crossing features. Each feature taken in isolation, an abnormal logon or an isolated device connection is not necessarily an indication of a threat. A weekend logon could be an indication of overtime or a late logon could be a diligent employee answering an email. However a late logon, abnormally large email attachment size, a URL to Dropbox not accessed before all in the previous 7 days may be an indication of a threat. Our model equally looks at these features from different permutations.

An abnormal email attachment does not in itself suggest a threat however if other features break out of their range, again this could be an indication of a threat. Figure 4.4 illustrates the class diagram design behind the Recursive feature builder.

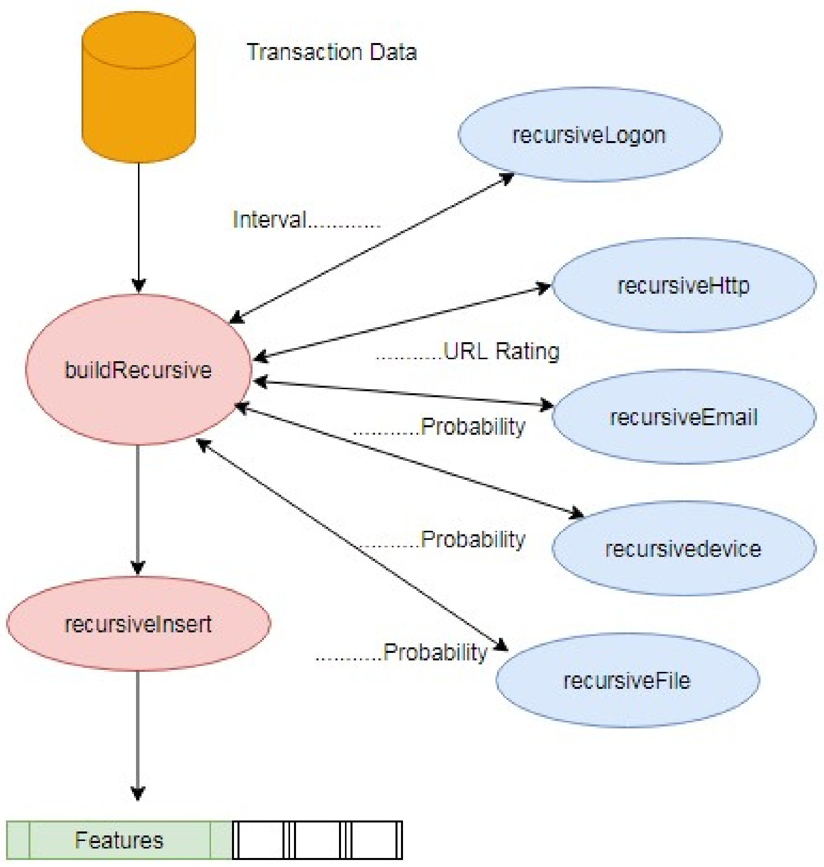


Figure 4.4 Recursive feature builder class diagram

BuildRecursive reads in each individual user activity from the transaction table. From the activity (logon, email ….. http) the decision is made what the makeup of that feature will be and what attributes will be applied to the classification. When user logs on to the system Buildrecursive will call each class in turn to return the value of the requested feature. Depending on that activity each other's action method will be called to contribute to the feature record. An interval parameter is passed to the method to define the data collection period.

RecursiveLogon	Return the probability of this logon's time from the population mean.
RecursiveDevice	Returns the probability of usage of that device connect in the interval period against the population of intervals.
RecursiveFile	Returns the probability of file access in the interval period against the population of intervals.
RecursiveEmail	Returns the probability of both number and attachment size of emails in the interval period against the population of intervals.
RecursiveHttp	Calculates the sum of the URL rating in the interval period.

Within our Machine Learning design we use a binary classification implemented with the Random Forest algorithm. Classification identifies which groups an action belongs to, threat or not a threat. Decision trees forms the basis of Random Forest and as implied based on questions traversing down a certain branch of the tree.
Decision trees provide scale and speed to the algorithm. Random Forest has a configuration of many trees working together. Each tree returns a prediction to the model and is scored to produce an overall prediction. The machine learning model is divided into 2 stages, training and testing. The dataset is split by percentage. Standard split is 70% used for training and 30% used for testing. Figure 4.5 illustrates the workflow stages of the ML process.

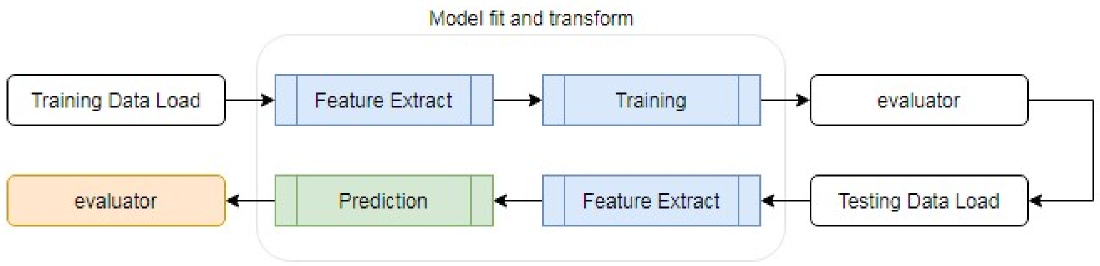


Figure 4.5 Machine Learning Workflow.

Additional configuration will be set within the algorithm and will be evaluated using different values to develop the most performant model. These parameters include seeding, impurity, depth, subsetstrategy, numtrees and maxbins. Each is explained in the implementation section.

The design of the testing module had to allow for scale, repetition of time period execution for testing and configuration change. In order to execute the model at given intervals a simple scheduler was developed which will execute each hour over a number of weeks or months. This allowed the evaluation of output at a given time. Each schedule only concerns the model with data up to that period. This allowed us to test single or all threat scenarios, stop the model and perform enhancements. Figure 4.6 illustrates how data is gathered for each test.
2 time slice dates are traversed each hour returning prediction at that instance in time.

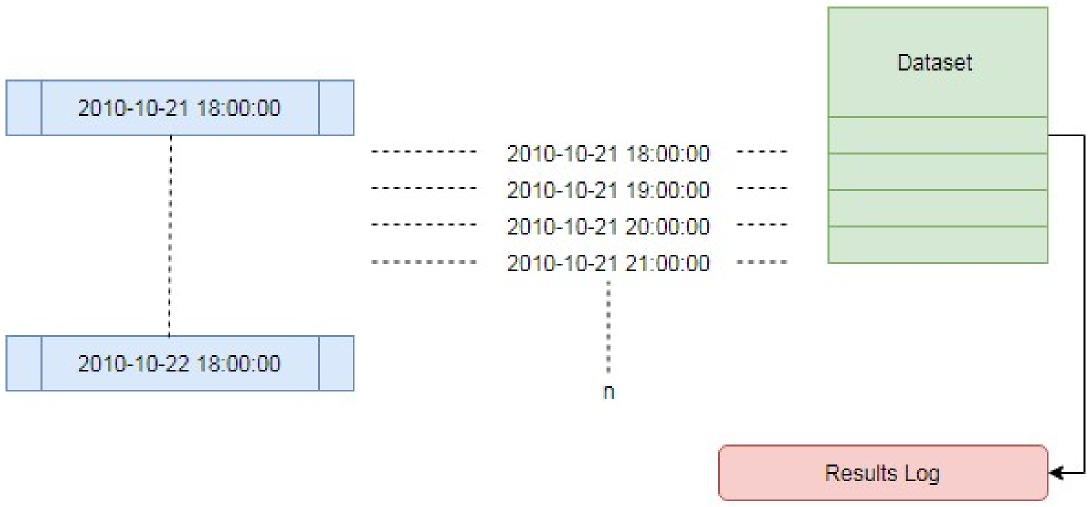
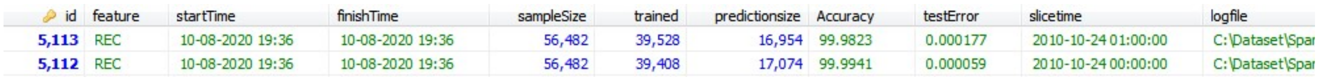


Figure 4.6 Test Period Scheduler.

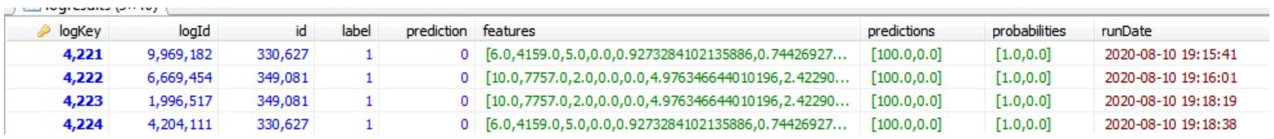
The application records 2 output log tables from the model.

1. A high level summary containing accuracy, test error and confusion matrix values for each time slice in the test



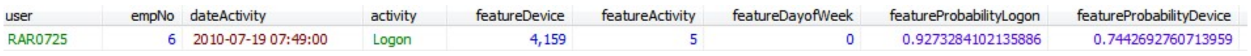
id	feature	startTime	finishTime	sampleSize	trained	predictionsize	Accuracy	testError	slicetime	logfile
5,113	REC	10-08-2020 19:36	10-08-2020 19:36	56,482	39,528	16,954	99.9823	0.000177	2010-10-24 01:00:00	C:\Dataset\Spar
5,112	REC	10-08-2020 19:36	10-08-2020 19:36	56,482	39,408	17,074	99.9941	0.000059	2010-10-24 00:00:00	C:\Dataset\Spar

2. A Detail of transactions which failed to predict correctly.



logKey	logId	id	label	prediction	features	predictions	probabilities	runDate
4,221	9,969,182	330,627	1	0	[6.0,4159.0,5.0,0.0,0.9273284102135886,0.74426927...	[100.0,0.0]	[1.0,0.0]	2020-08-10 19:15:41
4,222	6,669,454	349,081	1	0	[10.0,7757.0,2.0,0.0,0.0,4.976346644010196,2.42290...	[100.0,0.0]	[1.0,0.0]	2020-08-10 19:16:01
4,223	1,996,517	349,081	1	0	[10.0,7757.0,2.0,0.0,0.0,4.976346644010196,2.42290...	[100.0,0.0]	[1.0,0.0]	2020-08-10 19:18:19
4,224	4,204,111	330,627	1	0	[6.0,4159.0,5.0,0.0,0.9273284102135886,0.74426927...	[100.0,0.0]	[1.0,0.0]	2020-08-10 19:18:38

Details of the log results are then crossed with the features table to isolate which user transactions which failed the prediction.



user	empNo	dateActivity	activity	featureDevice	featureActivity	featureDayofWeek	featureProbabilityLogon	featureProbabilityDevice
RAR0725	6	2010-07-19 07:49:00	Logon	4,159	5	0	0.9273284102135886	0.7442692760713959

Finally, the application creates a csv file of all prediction records for all periods.

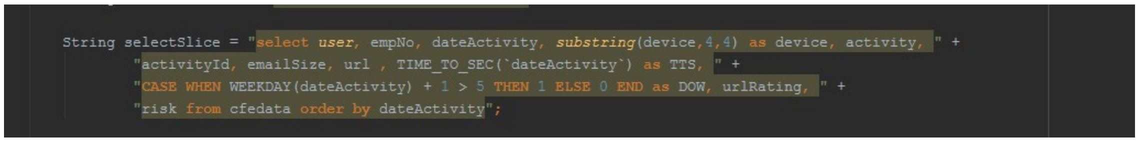
5 Implementation

The following section focuses on the technical implementation of the application which enabled us to begin testing our model. The section also outlines the series of tests completed, the logic behind the test and the desired outcomes.

The model takes each user activity and defines a footprint of that user's activity for a defined interval prior to the activity. This interval is parameterised in order to analyse if there is a different level of performance between selecting intervals. Intervals can be defined as x days / hours. A feature is then constructed from this action for the model.
The generic algorithm we named as Cross Feature Extraction with recursive feedback (CFERF). Cross feature refers to the algorithm extracting multiple probabilities to build features from a single activity; recursive as each activity is repeated and feedback as the algorithm can feedback the results as a feature to the next activity.

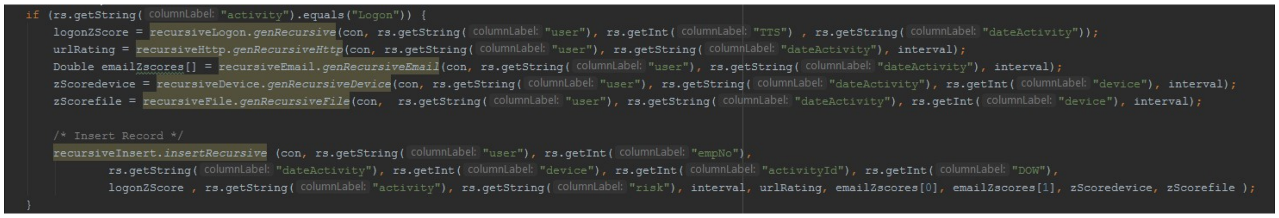
Features, logons, emails, device connections are calculated as outlined in the design specification based on their Z-score, measured from the population mean. URLs are rated with a score based on the severity of the site accessed and out of business days are flagged between 0/1 with 1 representing Saturday / Sunday.

Step 1 consists of all transactions getting selected from the source table for feature calculation.



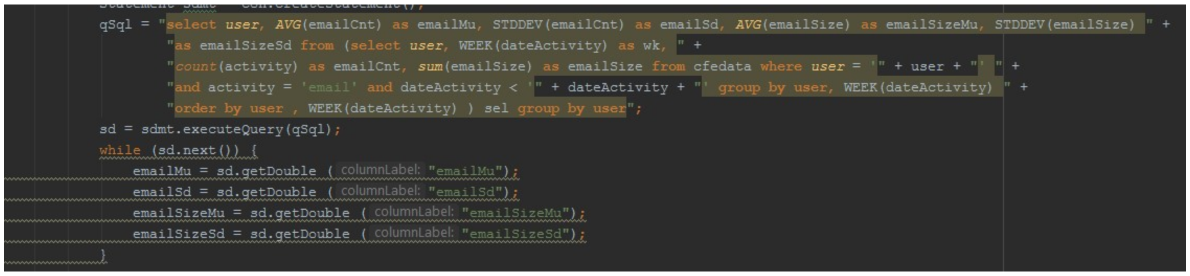
```
String selectSlice = "select user, empNo, dateActivity, substring(device,4,4) as device, activity, " +
        "activityId, emailSize, url , TIME_TO_SEC(`dateActivity`) as TTS, " +
        "CASE WHEN WEEKDAY(dateActivity) + 1 > 5 THEN 1 ELSE 0 END as DOW, urlRating, " +
        "risk from cfedata order by dateActivity";
```

For each activity type a method, passing the user, the interval required and the activity date is called to return z-scores.



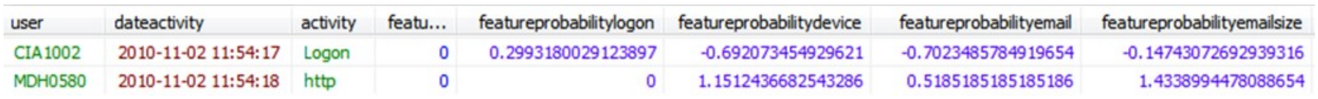
```
if (rs.getString( columnLabel: "activity").equals("Logon")) {
    logonZScore = recursiveLogon.genRecursive(con, rs.getString( columnLabel: "user"), rs.getInt( columnLabel: "TTS") , rs.getString( columnLabel: "dateActivity"));
    urlRating = recursiveHttp.genRecursiveHttp(con, rs.getString( columnLabel: "user"), rs.getString( columnLabel: "dateActivity"), interval);
    Double emailZscores[] = recursiveEmail.genRecursiveEmail(con, rs.getString( columnLabel: "user"), rs.getString( columnLabel: "dateActivity"), interval);
    zScoredevice = recursiveDevice.genRecursiveDevice(con, rs.getString( columnLabel: "user"), rs.getString( columnLabel: "dateActivity"), rs.getInt( columnLabel: "device"), interval);
    zScorefile = recursiveFile.genRecursiveFile(con,  rs.getString( columnLabel: "user"), rs.getString( columnLabel: "dateActivity"), rs.getInt( columnLabel: "device"), interval);

    /* Insert Record */
    recursiveInsert.insertRecursive (con, rs.getString( columnLabel: "user"), rs.getInt( columnLabel: "empNo"),
            rs.getString( columnLabel: "dateActivity"), rs.getInt( columnLabel: "device"), rs.getInt( columnLabel: "activityId"), rs.getInt( columnLabel: "DOW"),
            logonZScore , rs.getString( columnLabel: "activity"), rs.getString( columnLabel: "risk"), interval, urlRating, emailZscores[0], emailZscores[1], zScoredevice, zScorefile );
}
```

Within each method the mean and standard deviation for the given time slice interval is calculated. SQL calls back to the database with SQL functions AVG, STDDEV are used which are more efficient executing on the database rather than calculating in the java class.



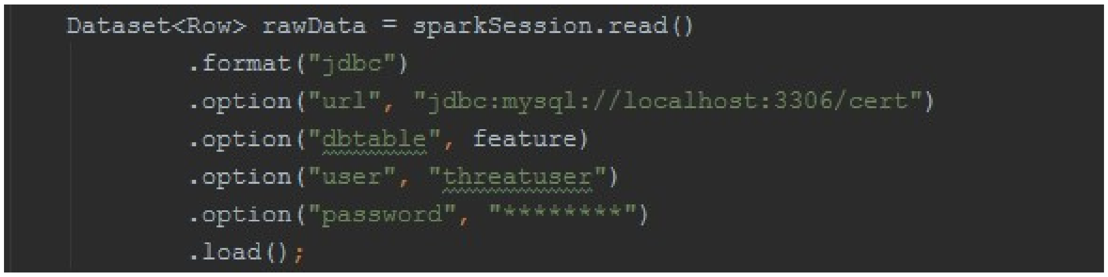
```
        qSql = "select user, AVG(emailCnt) as emailMu, STDDEV(emailCnt) as emailSd, AVG(emailSize) as emailSizeMu, STDDEV(emailSize) " +
                "as emailSizeSd from (select user, WEEK(dateActivity) as wk, " +
                "count(activity) as emailCnt, sum(emailSize) as emailSize from cfedata where user = '" + user + "' " +
                "and activity = 'email' and dateActivity < '" + dateActivity + "' group by user, WEEK(dateActivity) " +
                "order by user , WEEK(dateActivity) ) sel group by user";
        sd = sdmt.executeQuery(qSql);
        while (sd.next()) {
            emailMu = sd.getDouble ( columnLabel: "emailMu");
            emailSd = sd.getDouble ( columnLabel: "emailSd");
            emailSizeMu = sd.getDouble ( columnLabel: "emailSizeMu");
            emailSizeSd = sd.getDouble ( columnLabel: "emailSizeSd");
        }
```

The same logic is applied to the current time slice taken from the current dateActivity minus the interval period. Both are used in the standard z-score calculation to return the feature value.



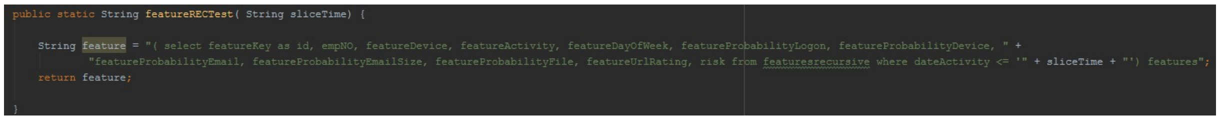
user	dateactivity	activity	featu...	featureprobabilitylogon	featureprobabilitydevice	featureprobabilityemail	featureprobabilityemailsize
CIA1002	2010-11-02 11:54:17	Logon	0	0.2993180029123897	-0.692073454929621	-0.7023485784919654	-0.14743072692939316
MDH0580	2010-11-02 11:54:18	http	0	0	1.1512436682543286	0.5185185185185186	1.4338994478088654

The development of the threat model was built using Apache Spark Machine Learning. As outlined, this gives the research a tool which can cater for the scale, speed and reliability required. Setting up the classifier is the first step of the machine learning algorithm. We load the dataset directly from the featuresrecursive table in the database through the spark session into a dataframe.



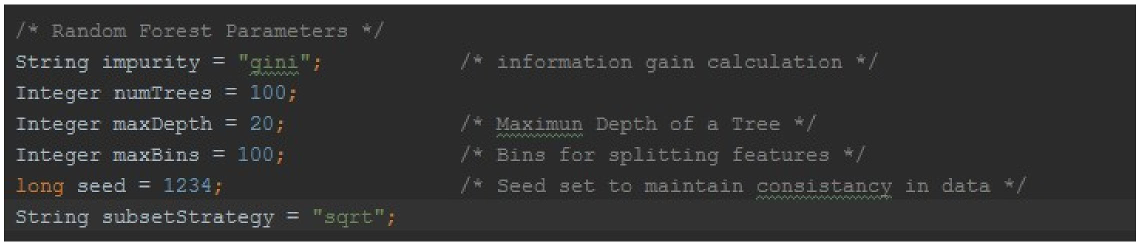
```
    Dataset<Row> rawData = sparkSession.read()
            .format("jdbc")
            .option("url", "jdbc:mysql://localhost:3306/cert")
            .option("dbtable", feature)
            .option("user", "threatuser")
            .option("password", "********")
            .load();
```

The feature variable from our FeatureModel method passes the time interval from the scheduler module. This enables the execution of multiple tests.



```
public static String featureRECTest( String sliceTime) {

    String feature = "( select featureKey as id, empNO, featureDevice, featureActivity, featureDayOfWeek, featureProbabilityLogon, featureProbabilityDevice, " +
            "featureProbabilityEmail, featureProbabilityEmailSize, featureProbabilityFile, featureUrlRating, risk from featuresrecursive where dateActivity <= '" + sliceTime + "') features";
    return feature;

}
```

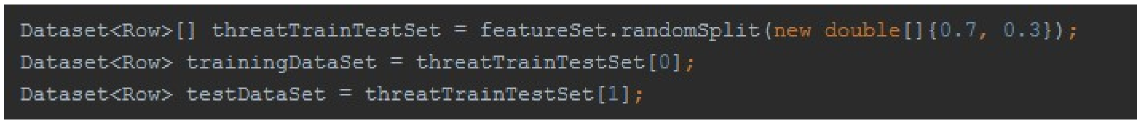
The dataset is then supplied with the header feature columns and the features label set from the risk field, 0 normal and 1 threat. The next implementation stage is to define the parameters (hyperparameters) for the machine learning algorithm. Initial configuration is set as follows:



```
/* Random Forest Parameters */
String impurity = "gini";            /* information gain calculation */
Integer numTrees = 100;
Integer maxDepth = 20;               /* Maximun Depth of a Tree */
Integer maxBins = 100;               /* Bins for splitting features */
long seed = 1234;                    /* Seed set to maintain consistancy in data */
String subsetStrategy = "sqrt";
```

Impurity gini:	this rates the chance of an incorrect classification.
numTrees:	set at an average rate to reduce error.
maxDepth:	sets the depth for each tree.
maxBins:	maximum value set for splitting features on each node.
seed:	random number, and repeats same output for same dataset.

The feature set is then split into the training and test dataset for presentation to the Machine Learning algorithm.



```
Dataset<Row>[] threatTrainTestSet = featureSet.randomSplit(new double[]{0.7, 0.3});
Dataset<Row> trainingDataSet = threatTrainTestSet[0];
Dataset<Row> testDataSet = threatTrainTestSet[1];
```

The random Forest classifier is built with the setup parameters and fit with the training dataset. A new predictions dataset is then created from transforming the test dataset. Finally using a multi classification evaluator and a multi class metric evaluator we can extract accuracy, test error and confusion matrix results.

Our attack vector consists of a number of malicious activities in CERT data in addition to synthetic transaction generated from our application. These additional transactions can be scaled up where required. CERT dataset consists of 3 scenarios

- Users begin connecting to removable devices after hours and uploading information to WikiLeaks.
- Users accessing job websites and begin connecting to removable devices at abnormal rates.
- System Administrator downloads KeyLogger software.

Because of the imbalance of threats and the lack of diversity in threat types within the dataset we added in a number of generated synthetic threats. Synthetic activities were implemented by replicating existing user profiles and adding additional transactions. We add additional

- Logon times outside the normal working.
- Emails with larger than normal attachments.
- Device connects greater than normal activity.

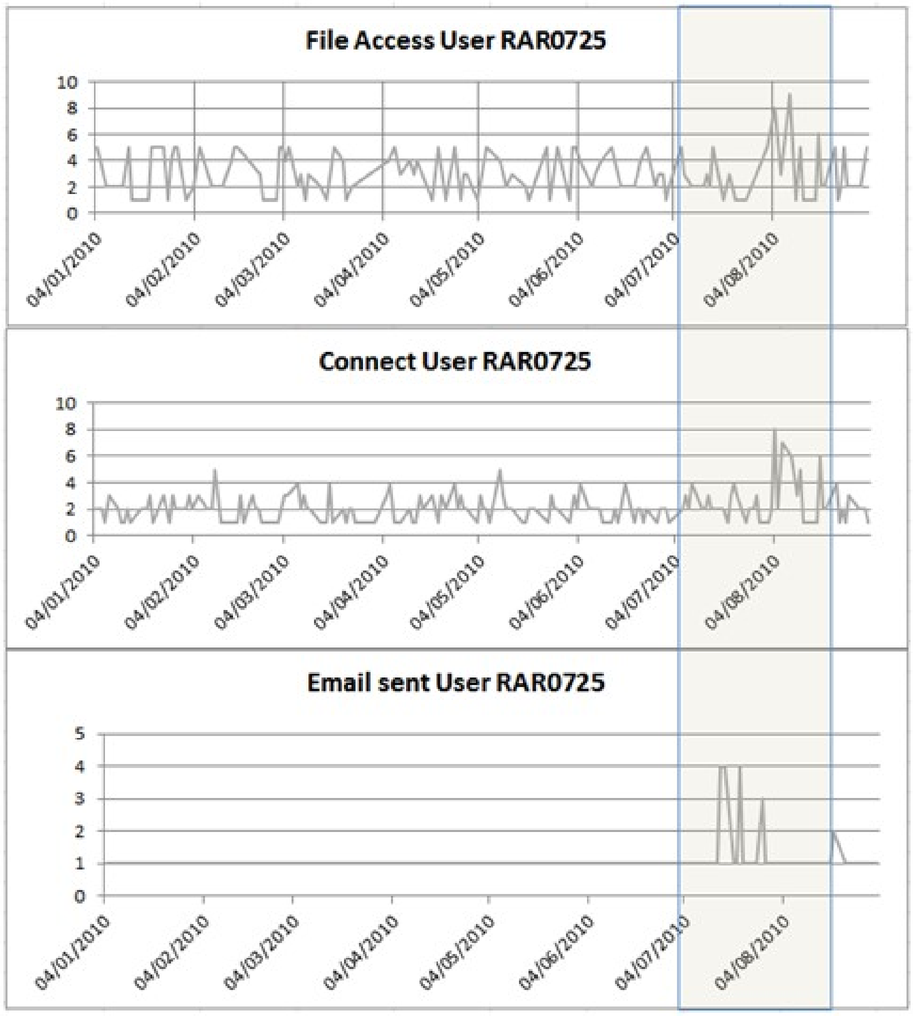
Table 5.1 shows 4 of the specific use cases of threat.

scenario	User	Threat Date	Activity	Threat description
CERT 1	AAM0658	23/10/2010 06:26	http	Accessed http://wikileaks.org Normal PC, no Abnormal logon.
CERT 2	RAR0725			Resignation 19th Aug, High level of device connects 6th Aug.
Synthetic	CIA1001	23/10/2010 22:20	Logon	Late logon and 3 large email previous week plus url accessed containing espionage
Synthetic	CIA1002	21/10/2010 20:01	email	Large Email with 15 connects previous day

Table: 5.1 Threat use cases.

We take use case RAR0725, Figure 5.1 and have defined that over the 4-8 Aug that this is a suspicious threat frame. The user remains in the organisation until 19th of August of that year and then resigns.

Figure 5.1 User RAR0725 system activity over 8 months



When we analyse the data set against the probability output for user RAR0725 and we see same patterns as outlined in Figure 5.2

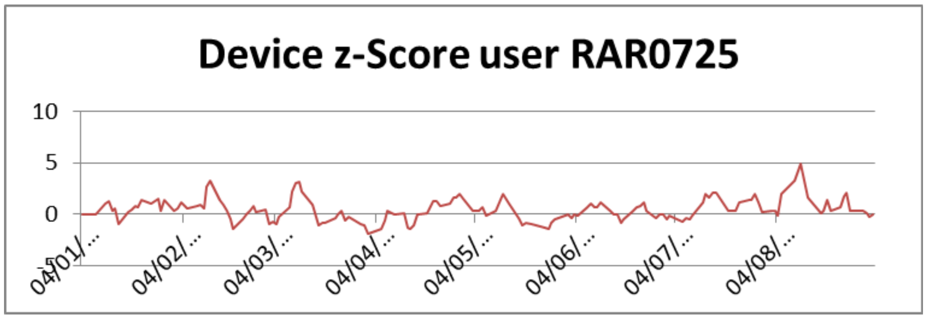
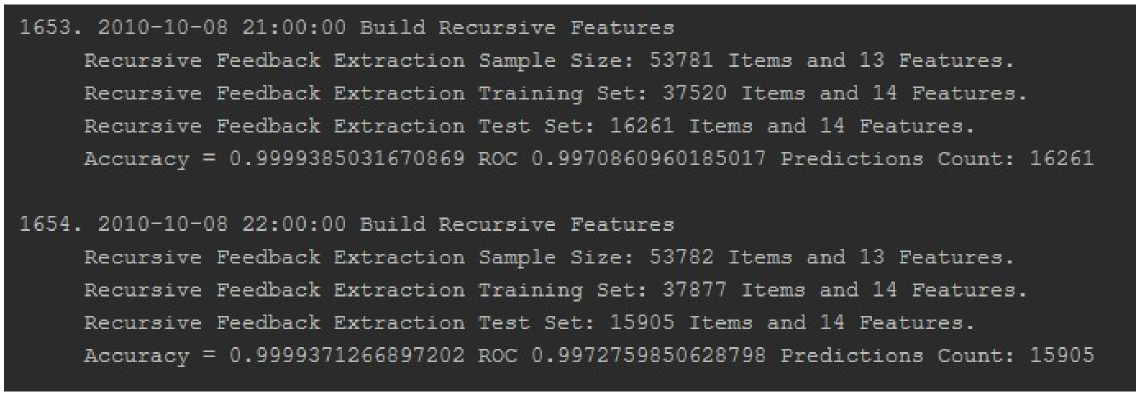


Figure 5.2 User RAR0725 probability activity over 8 months

The focus of our research is whether our model can offer data in order to pre-empt a potential threat or make a decision to take action against the activity. As a user logs on to the system, by this action can the model supply the predictive data for the system to restrict the user. Both the restriction itself and the speed of the predictive calculation are beyond the scope of this paper. However, in order to supply the prediction data our model must execute in real time (Figure 5.3). In order to achieve this we stream datasets at the model. We also compare different stages in time, a scheduler was developed which can be set to execute simulating time intervals. The model was run on each hour over a period of 12 months to obtain accuracy as threats arrived into the dataset.



```
1653. 2010-10-08 21:00:00 Build Recursive Features
     Recursive Feedback Extraction Sample Size: 53781 Items and 13 Features.
     Recursive Feedback Extraction Training Set: 37520 Items and 14 Features.
     Recursive Feedback Extraction Test Set: 16261 Items and 14 Features.
     Accuracy = 0.9999385031670869 ROC 0.9970860960185017 Predictions Count: 16261

1654. 2010-10-08 22:00:00 Build Recursive Features
     Recursive Feedback Extraction Sample Size: 53782 Items and 13 Features.
     Recursive Feedback Extraction Training Set: 37877 Items and 14 Features.
     Recursive Feedback Extraction Test Set: 15905 Items and 14 Features.
     Accuracy = 0.9999371266897202 ROC 0.9972759850628798 Predictions Count: 15905
```

Figure 5.3 Scheduled time intervals of the model.

In implementing the test stage of the research we outline 3 questions:

- What are the experiments to be undertaken?
- What metrics are best to evaluate the outcome of the experiments?
- How does the evaluation contribute to the research question?

To analyse the effectiveness and present results of our recursive algorithm, we take a sample of user's activity from the dataset with 5 different threat scenario types and measure against accuracy, test error and confusion matrix. We use the confusion matrix to ascertain the number of correctly predicted cases and the level of false positives. Finally we adjust the Hyperparameters within and test for performance.

6 Evaluation

To evaluate the research question put forward in this paper, a series of experiments was performed against our 2 developed models: Cross feature extraction with recursive feedback and a simple individual features model. Statistical analysis of the output is averaged and summarised below.

6.1 Actual Activity versus Features

Before we run a series of experiments we want to see if there is consistency in our transformation from actual activity to features presented to the machine learning algorithm.

Outlined in Figure 6.1 we take 2 of the threat users and compare their device connections and email attachment size with the Z-Score calculated for the features.

User	Threat Date	Activity	Threat target description
RAR0725	2010-08-07 07:49	Logon	Logon after high levels of device connection.
CIA1001	2010-10-23 22:20	Logon	After business logon, large email attachments sent in the previous days, suspicious website accessed containing espionage details.

Figure 6.1 Threat scenarios.

User RAR0725 shows a consistent pattern in figure 6.2 between actual connections to devices and Z-Scores for the given time frame.

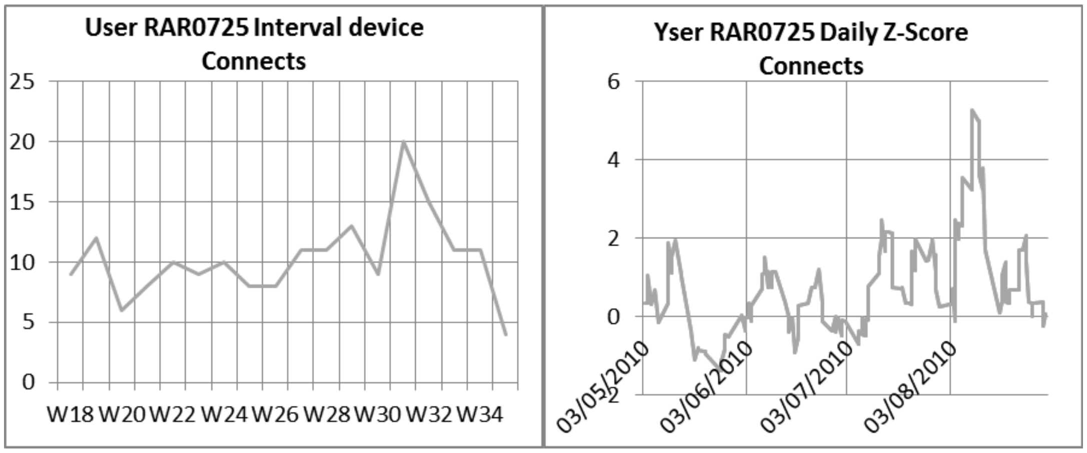


Figure 6.2 User RAR0725 device connection numbers by week against Z-Scores

User CIA1001 shows in figure 6.3 a similar patterns in the size of email attachments which was sent out by the user.

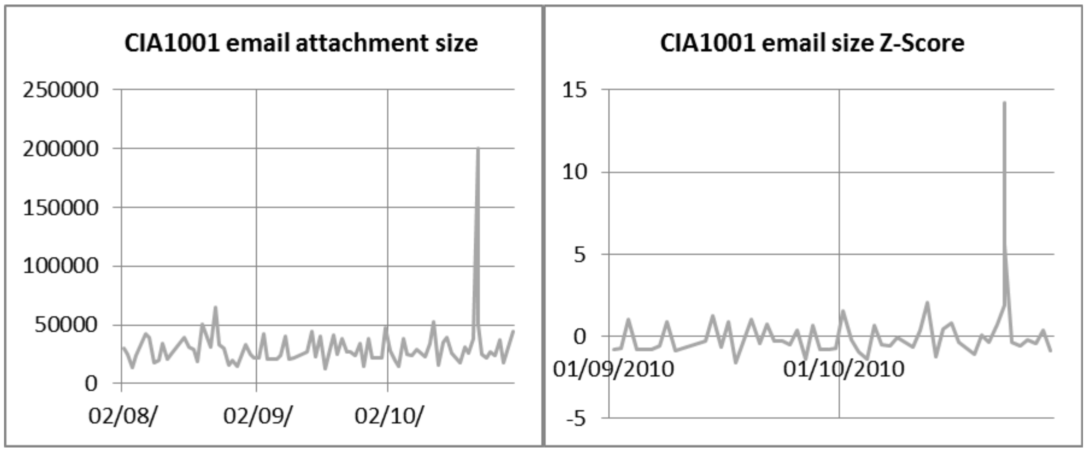


Figure 6.3 User CIA1001 email attachment size by week against Z-Scores.

If we expand this to the entire data population and take a third scenario of abnormal logon times and compare the general population with the threat population. What we find is, overall the pattern is similar between non threat and threat population. Figure 6.4 shows logon times converted to seconds for the non-threat population over a period of 17 months. Figure 6.5 shows the same data for the threat population.

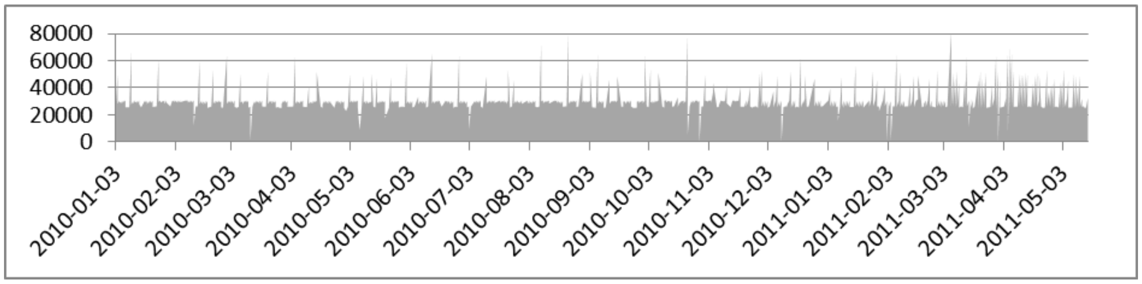


Figure 6.4 Non threat population logon time.

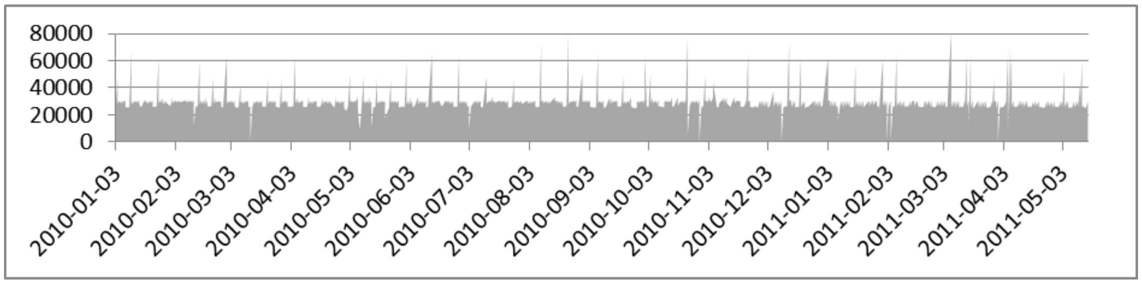
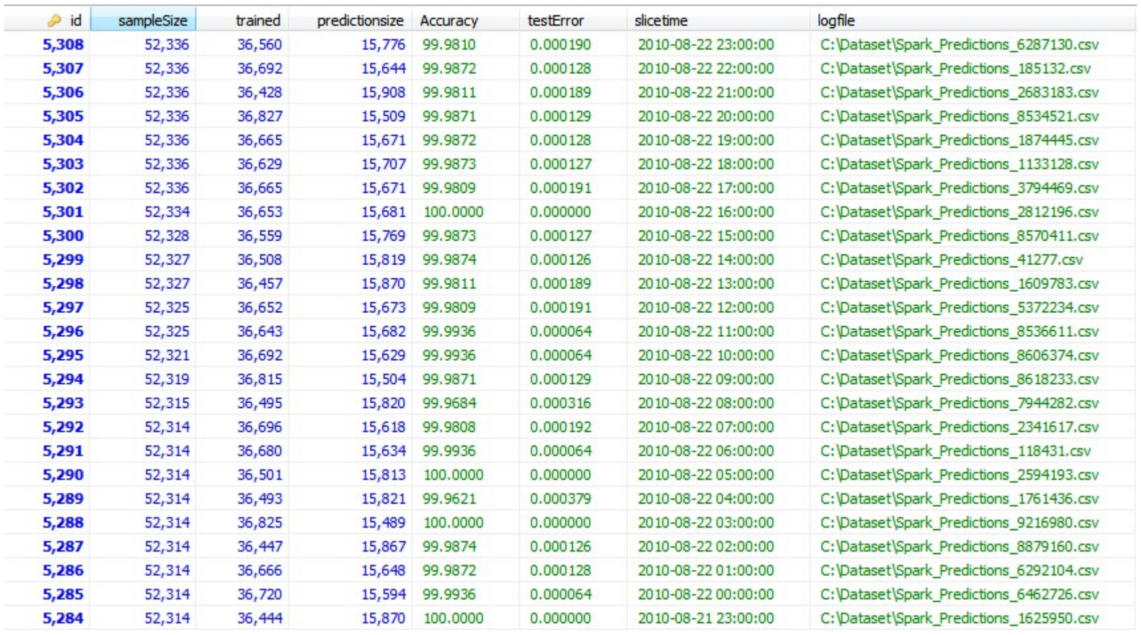


Figure 6.5 Threat population logon time.

This forms the basis of our thinking that single scenario use cases alone is not an indication of a threat, that abnormal system access as shown above is a common feature in the work environment. As we will show in experiment 6.4 detection of abnormality for a single feature is highly performant, however as a general profile within the population this will put additional pressure on referrals to security operators.

6.2 Cross Feature Extraction

This begins to build the first case for our algorithm that activities within the dataset alone are not an indication of a threat. Our algorithm crosses the features checking if there is a lower rate of false positives and false negatives. The following experiments have been performed. We initially run the model for a continuous week without any labelled threats in that period or the prior 3 weeks. 168 tests will run from with a dataset from 3rd January to 22nd of August. The test period will be hourly from the 16th to 22nd August. We have configured a 7 days interval for the algorithm. For each activity the model will analyse and add addition features for the previous interval (7 days) against every interval in the users history..
Figure 6.6 shows details of the output log table containing accuracy, test error, run time and log file output for the 166 tests.



id	sampleSize	trained	predictionsize	Accuracy	testError	slicetime	logfile
5,308	52,336	36,560	15,776	99.9810	0.000190	2010-08-22 23:00:00	C:\Dataset\Spark_Predictions_6287130.csv
5,307	52,336	36,692	15,644	99.9872	0.000128	2010-08-22 22:00:00	C:\Dataset\Spark_Predictions_185132.csv
5,306	52,336	36,428	15,908	99.9811	0.000189	2010-08-22 21:00:00	C:\Dataset\Spark_Predictions_2683183.csv
5,305	52,336	36,827	15,509	99.9871	0.000129	2010-08-22 20:00:00	C:\Dataset\Spark_Predictions_8534521.csv
5,304	52,336	36,665	15,671	99.9872	0.000128	2010-08-22 19:00:00	C:\Dataset\Spark_Predictions_1874445.csv
5,303	52,336	36,629	15,707	99.9873	0.000127	2010-08-22 18:00:00	C:\Dataset\Spark_Predictions_1133128.csv
5,302	52,336	36,665	15,671	99.9809	0.000191	2010-08-22 17:00:00	C:\Dataset\Spark_Predictions_3794469.csv
5,301	52,334	36,653	15,681	100.0000	0.000000	2010-08-22 16:00:00	C:\Dataset\Spark_Predictions_2812196.csv
5,300	52,328	36,559	15,769	99.9873	0.000127	2010-08-22 15:00:00	C:\Dataset\Spark_Predictions_8570411.csv
5,299	52,327	36,508	15,819	99.9874	0.000126	2010-08-22 14:00:00	C:\Dataset\Spark_Predictions_41277.csv
5,298	52,327	36,457	15,870	99.9811	0.000189	2010-08-22 13:00:00	C:\Dataset\Spark_Predictions_1609783.csv
5,297	52,325	36,652	15,673	99.9809	0.000191	2010-08-22 12:00:00	C:\Dataset\Spark_Predictions_5372234.csv
5,296	52,325	36,643	15,682	99.9936	0.000064	2010-08-22 11:00:00	C:\Dataset\Spark_Predictions_8536611.csv
5,295	52,321	36,692	15,629	99.9936	0.000064	2010-08-22 10:00:00	C:\Dataset\Spark_Predictions_8606374.csv
5,294	52,319	36,815	15,504	99.9871	0.000129	2010-08-22 09:00:00	C:\Dataset\Spark_Predictions_8618233.csv
5,293	52,315	36,495	15,820	99.9684	0.000316	2010-08-22 08:00:00	C:\Dataset\Spark_Predictions_7944282.csv
5,292	52,314	36,696	15,618	99.9808	0.000192	2010-08-22 07:00:00	C:\Dataset\Spark_Predictions_2341617.csv
5,291	52,314	36,680	15,634	99.9936	0.000064	2010-08-22 06:00:00	C:\Dataset\Spark_Predictions_118431.csv
5,290	52,314	36,501	15,813	100.0000	0.000000	2010-08-22 05:00:00	C:\Dataset\Spark_Predictions_2594193.csv
5,289	52,314	36,493	15,821	99.9621	0.000379	2010-08-22 04:00:00	C:\Dataset\Spark_Predictions_1761436.csv
5,288	52,314	36,825	15,489	100.0000	0.000000	2010-08-22 03:00:00	C:\Dataset\Spark_Predictions_9216980.csv
5,287	52,314	36,447	15,867	99.9874	0.000126	2010-08-22 02:00:00	C:\Dataset\Spark_Predictions_8879160.csv
5,286	52,314	36,666	15,648	99.9872	0.000128	2010-08-22 01:00:00	C:\Dataset\Spark_Predictions_6292104.csv
5,285	52,314	36,720	15,594	99.9936	0.000064	2010-08-22 00:00:00	C:\Dataset\Spark_Predictions_6462726.csv
5,284	52,314	36,444	15,870	100.0000	0.000000	2010-08-21 23:00:00	C:\Dataset\Spark_Predictions_1625950.csv

Figure 6.6 Accuracy log output.

Average performance over 168 tests							
Tests	Sample size	Trained size	Prediction size	Accuracy	Test error	Min accuracy	Max accuracy
168	51,734	36,203	15,531	99.98%	0.000140	99.94%	100%

Figure 6.7 Average accuracy levels.

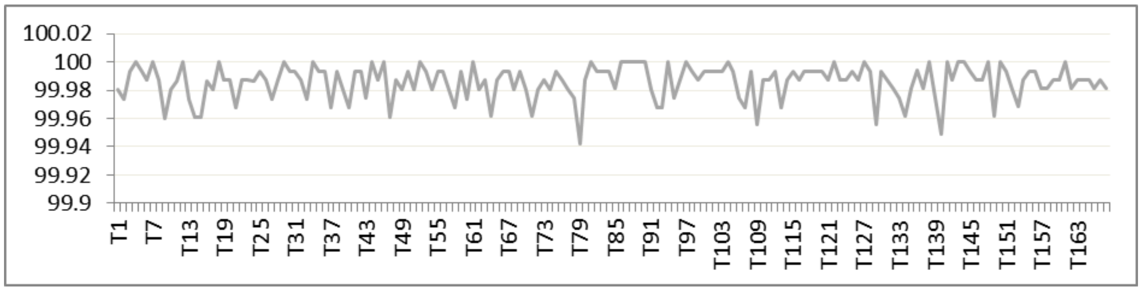


Figure 6.8 Accuracy Levels over 168 test probes.

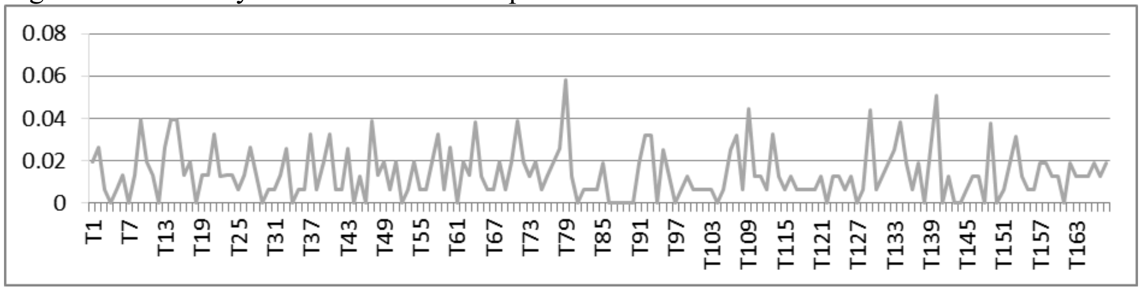


Figure 6.9 Test error Levels over 168 test probes.

We get an overall predictive average of 99.98% accuracy illustrated in figure 6.7. This is an positive initial result and we see consistency of this level of accuracy across all test probes illustrated in figure 6.9. With a small number of threats in this period and breakout levels against the mean within range we expect to see the machine learning algorithm to produce this.

6.3 Cross Feature Extraction with Multiple Threats

We take the period containing the most threats and compare our results from our previous experiment. We will expand the matrix to give more performance detail on the output.

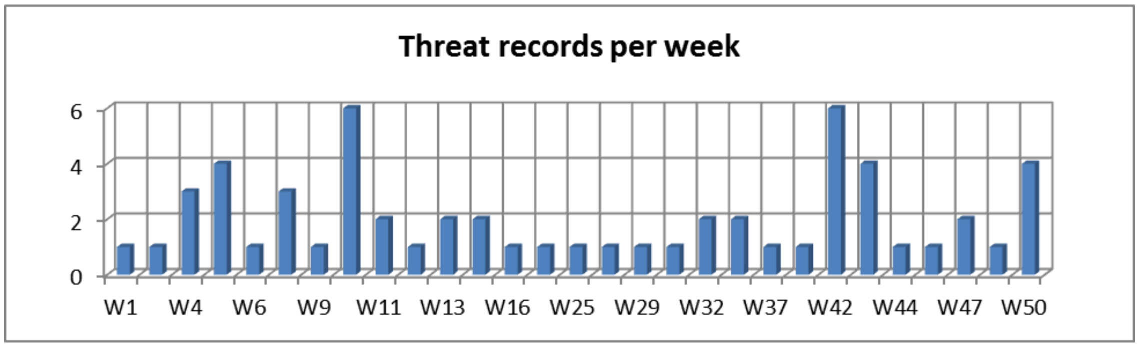


Figure 6.10 Number of threat records per week.

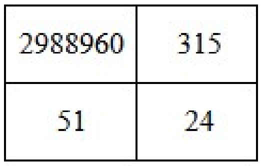
We change our schedule test dates from 21 October to 27 October where we encounter 8 separate threats outlined in figure 6.13.

Prediction Performance Measure Output						
Tests	Description	Sample size	Trained size	Prediction size	Accuracy	Test error
150	Average	66,331	46,424	19,907	99.984%	0.000146

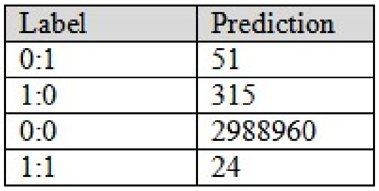
1	Oct-27 23:00	67,177	47,044	20,133	99.995%	0.000050

Figure 6.11 Accuracy levels for threat period.

Figure 6.11 shows the output of the model for a period with 8 threats. Again we see high levels of accuracy and consistency of that accuracy over the test period. We use a confusion matrix for our binary classification to analyse the overall performance of the model. Over 150 tests were run, one each hour in a normal week with an overall total of 2,989,350 observations. The classifier model gets 2,988,960 correct where the activity is labelled normal activity and is actually normal and not threats (True negatives). The model predicts a further 24 threats which are actually threats (True Positives). Overall the model predicts 2,998,984 correct predictions with 366 incorrect giving an accuracy of 99.98%. Figure 6.12 outline the detail.



2988960	315
51	24



Label	Prediction
0:1	51
1:0	315
0:0	2988960
1:1	24

Figure 6.12 Confusion Matrix.

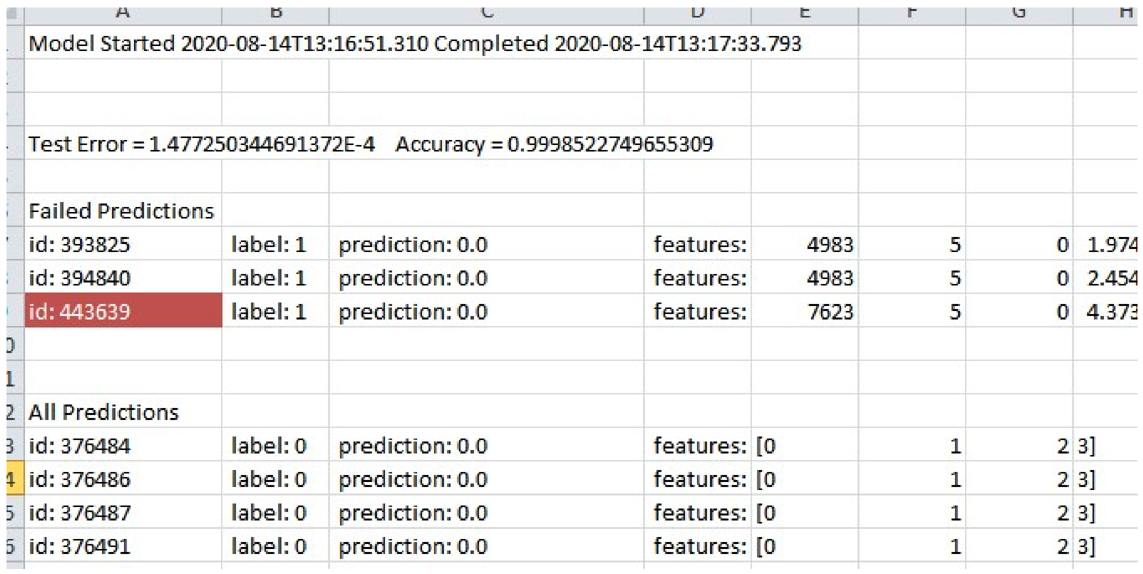
From the opposite squares in the matrix we see that the model predicted 315 false positives where threats not being caught and 51 false negatives, normal active predicted as threats.

However this strengthens our approach. Our model takes an instance in time and looks back over an interval period for abnormality, in this test case 7 days. We expect false positives to be high as these were threats from previous months and at our run date we are not expecting the model to always predict them. We need to ascertain if our model predicted threats in that 7 day period alone. In the 7 day period 8 threats existed as shown in Figure 6.13.

featureKey	User	Activity Date	Activity	Risk
442349	CIA1002	21/10/2010 20:01	email	Threat
442662	CIA1001	22/10/2010 19:20	file	Threat
442663	CIA1001	22/10/2010 19:20	http	Threat
442679	TAP0551	23/10/2010 04:53	http	Threat
442682	AAM0658	23/10/2010 06:26	http	Threat
442706	CIA1001	23/10/2010 22:20	Logon	Threat
443639	TAP0551	27/10/2010 21:09	Logon	Threat
443656	WDV0573	27/10/2010 22:26	Logon	Threat

Figure 6.13 threats between 21 and 27 October.

The model failed to predict featureKey 443639 however predicted the other 7 threats correctly. Figure 6.14 shows the individual failed prediction records.



```
Model Started 2020-08-14T13:16:51.310 Completed 2020-08-14T13:17:33.793


Test Error = 1.477250344691372E-4   Accuracy = 0.9998522749655309

Failed Predictions
id: 393825   label: 1   prediction: 0.0   features:   4983   5   0   1.974
id: 394840   label: 1   prediction: 0.0   features:   4983   5   0   2.454
id: 443639   label: 1   prediction: 0.0   features:   7623   5   0   4.373


All Predictions
id: 376484   label: 0   prediction: 0.0   features: [0   1   2 3]
id: 376486   label: 0   prediction: 0.0   features: [0   1   2 3]
id: 376487   label: 0   prediction: 0.0   features: [0   1   2 3]
id: 376491   label: 0   prediction: 0.0   features: [0   1   2 3]
```

Figure 6.14 All predictions output.

6.4 Experiment Single feature.

In this experiment we want to test a single scenario to ascertain the performance. We developed a second algorithm to detect single activity use cases. We take the case of an abnormal logon to the system. The features are limited to abnormal logon time and day of week.
We want to compare this approach to evaluate the potential number of true positives which in turn would have to be directed towards the security operations center for review.

Sample size	Trained size	Prediction size	Accuracy	Test error
5014	3525	1489	99.86%	0.001343

0:1	2
1:0	0
0:0	1477
1:1	10

Figure 6.15 Accuracy levels and confusion matrix.

Figure 6.15 shows the result with high level of accuracy and all 10 abnormal threats caught. 2 cases of normal behavior were labeled as threats. This does indicate success for the model however as we do not see the depth to the approach. Scenarios in isolation will always have to be referred to expensive security operations.

6.5 Tuning Hyperparameters

Having produced output from the model, we consider the impact of tuning the model with hyperparameters. We run a single test on a sample size of 89,051. Initially we take numtrees, the most important configuration parameter and the maxDepth maximum level for each tree. The default setting for our test so far in this research is 100 and this has produced accuracy levels of 99.992%. Reducing the numTrees by 50% and increasing the maxDepth by 25% has a different in the level of accuracy from .992 to .996 resulting in reducing failed predictions from 2 to 1 as outlines in figure 6.16. With this dataset the indications are the model performs better with reduced tress but increased depth using impurity gini. Changing to entropy caused an additional failed prediction in the same dataset.

Impurity measure	numTrees	maxDepth	Subset strategy	Sample size	Accuracy	Test error	Failed Predictions
gini	100	20	sqrt	89,051	99.992	0.000074	2
gini	50	25	sqrt	89,051	99.996	0.000037	1
entropy	100	20	sqrt	89,051	99.988	0.000112	3
entropy	50	25	sqrt	89,051	99.992	0.000075	2

Figure 6.16 Hyperparameters configuration.

We take a look at the training / testing split ratio. 70:30 is the recommended value for balance. We test this on our dataset and find this to be the case. As we reduce the level of training transactions Figure 6.17 shows a reduction in the overall level of accuracy supplied by the model.

Split	Sample size	Accuracy	Test error
70:30	89,051	99.992	0.000074
60:40	89,051	99.988	0.000112
50:50	89,051	99.986	0.000134

Figure 6.17 Training / Testing split ratio.

6.6 Discussion

The results coming from our Random Forest classifier model, overall is showing positive results with close to 100% accuracy. The concern is with this level of imbalanced data that close to 100% would have been expected. In the example period, the model managed to detect 7 out of 8 threats but it is apparent that an understanding of the threat use case is still required. It does suggest that expanding the activities from logon, email, device, file access, http to other business activities such as report generation, client access has merit. The concept of checking each activity against the footprint of a previous day, week would be the same. However because of the complexity of use cases in such are large dataset, as a minimum this would not be feasible without automated testing. The approach in this research has proven that in this specific use cases performance if high but there is a concern around the logic of an interval. For example if a user consistently loaded small amounts of data on the last day of each month this model is unlikely to predict that. However extending the activity range to the source of that data may add value. Unsupervised learning may also be an option in addressing the activity scope constraint however as seen from previous research this is at the cost of accuracy. For our experiment we injected a number of threat scenarios into the data population however these were specific to prove the model approach. Additional inconsistent threat scenarios should also be included. The experiment allowed model to consider relationships between activities but at a generic level. More analysis is required on the footprint of a threat to fully test the boundary of the model.

7 Conclusion and Future Work

We outlined throughout this paper the challenges facing corporate and government organisations to protect sensitive data from insider threats.

The objective of this research was to analyse, if providing features with enhanced relationships to machine learning algorithms had the potential of providing high levels of accuracy. There were 3 parts set out in defining this research question, enhancing user features, pre-emptive and detection.

We provided an approach to enhancing the feature preparation, extracting across features and calculating a probability rating for the user activity merged with previous activity over defined intervals. Based on our experiments, a high level of accuracy and threat detection has been achieved. A pre-emptive notion has been partially achieved. Based on our experiments this paper has shown the ability to detect insider threat activity within the dataset. The detection has to assume either damage could have been done or was pre-empted by the detection.

We acknowledge that patterns existed within the dataset and even with the addition of extra synthetic threat scenarios; this may not always reflect the situation in the real production environments. Equally the speed and complexity of large multi-application environments presenting data to model needs further research.
We identify future areas of work:

1. Our approach was to develop a generic algorithm in relation to a fixed number of activities and fixed intervals. Further research would be interesting in relation to undefined intervals and extending the concept to application modules, accounting, reporting, client access, to research if the same model holds.

2. Because of volume of data required and the run time taken to test the models, research into how cloud solutions could provide a more advanced test environment which would be of interest.

References

Cole, E. SANS Institute. (2017). Defending against the wrong enemy: 2017 sans insider threat survey: pp 7.

Ponemon Institute and IBM. (2018). Data Breach Study: Impact of Business Continuity Management. pp 10.

Garcia-Gil, D. and Ramirez-Gallego, S. and Garcia, S. and Herrera, F. (2017). A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. pp 9.

Theoharidou, M. and Kokolakis S. and Karyda M. and Kiountouzis and E. (2005). The insider threat to information systems and the effectiveness of iso17799., Computers Security, 24(6) pp. 472-484.

Dawn, M. and Cappelli, A. Moore, P. and Trzeciak, R. F. (2012). The cert guide to insider threats: How to prevent, detect, and respond to information technology crimes (theft, sabotage, fraud).

Yang, G. and Cai, L. and Yu, A. and Ma, J. and Meng, D. (2018). Potential malicious insiders detection based on a comprehensive security psychological model, Proceedings of the 2017 International Workshop on Managing Insider Security Threats. pp. 9-16.

Maasberg, M. and Warren, J. and Beebe, N.L. (2015). The dark side of the insider: Detecting the insider threat through examination of dark triad personality traits, 2015 48th Hawaii International Conference on System Sciences. pp. 3518-3526.

Sood, A. K. and Zeadally, S. and Bansal, R. (2017). Exploiting trust: Stealthy attacks through socioware and insider threats, IEEE Systems Journal Year: 2017 , Volume: 11 , Issue: 2. pp. 415-426.

Youli, H. and Chao, L. (2014). A comparative study between the dark triad of personality and the big five, Canadian Social Science Vol. 11, No. 1, 2015. pp. 93-98.

Jiang, J. and Chen, J. and Choo, K. R. and Liu, K. and Liu, C. and Yu, M. and Mohapatra, P. (2018). Prediction and detection of malicious insiders' motivation based on sentiment profile on webpages and emails, MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM). pp. 1-6.

Bulling, D. and Scalora, M.J, and Borum, R. and Panuzio, J.(2008). Behavioral Science Guidelines for Assessing Insider Threats. pp. 9-11.

Legg, P. A. and Buckley, O. and Goldsmith, M. and Creese, S. (2017). Automated insider threat detection system using user and role-based profile assessment, IEEE Systems Journal Year: 2017, Volume: 11, Issue: 2. pp. 503-512.

Meng, F. and Lou, F. and Fu, Y. and Tian, Z. (2018). Deep learning based attribute classification insider threat detection for data security, 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC). pp. 576-581.

Tuor, A. and Kaplan, S. and Hutchinson, B. and Nichols, N. and Robinson, S. (2017). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. pp. 9.

Jang-Jaccard, J. and Nepal, S. (2012). A survey of emerging threats in cybersecurity, Journal of Computer and System Sciences. pp. 14-15.

Le, D. C. and Zincir-Heywood, A. N. (2018). Evaluating insider threat detection workflow using supervised and unsupervised learning, 2018 IEEE Security and Privacy Workshops (SPW). pp. 270-275.

Zaytsev, A. and Malyuk, A. and Miloslavskaya, N. (2017). Critical analysis in the research area of insider threats, 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud). pp. 288-296.

Hall, A.J. and Pitropakis, N. and Buchanan, W.J. and Moradpoor, N. (2018). Predicting Malicious Insider Threat Scenarios Using Organizational Data and a Heterogeneous Stack-Classifier. pp. 4-5.

Zhang, C. and Bengio, S. and Hardt, M. and Recht, B. and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization (ICLR 2017).