International Journal of Cyber Warfare and Terrorism
Volume 10 • Issue 2 • April-June 2020

Insider Threat Detection Using Supervised Machine Learning Algorithms on an Extremely Imbalanced Dataset

Naghmeh Moradpoor Sheykhkanloo, Edinburgh Napier University, Edinburgh, UK
Adam Hall, Edinburgh Napier University, Edinburgh, UK

ABSTRACT

An insider threat can take on many forms and fall under different categories. This includes malicious insider, careless/unaware/uneducated/naïve employee, and the third-party contractor. Machine learning techniques have been studied in published literature as a promising solution for such threats. However, they can be biased and/or inaccurate when the associated dataset is hugely imbalanced. Therefore, this article addresses the insider threat detection on an extremely imbalanced dataset which includes employing a popular balancing technique known as spread subsample. The results show that although balancing the dataset using this technique did not improve performance metrics, it did improve the time taken to build the model and the time taken to test the model. Additionally, the authors realised that running the chosen classifiers with parameters other than the default ones has an impact on both balanced and imbalanced scenarios, but the impact is significantly stronger when using the imbalanced dataset.

KEYWORDS

Data Pre-Processing, Imbalanced Dataset, Insider Threat, Spread Subsample, Supervised Machine Learning

1. INTRODUCTION

Insider attacks present a considerable issue in the cyber-threat landscape, with 40% of organisations labelling the vector as the most damaging attack faced (Cole, 2017) and (Moradpoor, 2017). In 2016, the containment and remediation of reported insider threats cost affected organisations 4 million dollars on average (Ponemon Institute, 2016). In addition, insider threats are extremely common among cyber-incidents; in 2015, 55% of cyber-attacks were insider threat cases (Bradley, 2015). Despite the high cost and frequent occurrence of insider threat attacks, detection and mitigation remain a problem. In 2018, 90% of companies are regarded vulnerable (Insiders, 2018). A further 38% of companies acknowledge that their insider threat detection and prevention capabilities are not adequate (Cole, 2017). This disparity demonstrates a significant gap between the current advancements in insider threat detection, and the requirements of businesses. Given the availability of computational resources, it is feasible to use Machine Learning (ML) techniques to solve problems of larger complexity than has previously been possible. A strong precedent of this can be observed in recent history with the growth of the field of Big Data. This is also exemplified by the historic achievement of Google Deepmind (Hassabis, 2017), creating a machine learning algorithm which masters the immensely complex board game Go (Silver, 2016). Most organisations have the resources to keep logs of employee interactions with technology. By harnessing the data produced through logging, this information could be digested

DOI: 10.4018/IJCWT.2020040101

into a format upon which predictions regarding insider threat cases could be made. Having said this, a data driven approach to insider threat mitigation is not a new idea, this is a field experiencing an increasing rate of publication. However, vanguard attempts still report more effective models than later cases where machine learning has been applied (Gheyas, 2016).

In machine learning/data mining projects, an imbalanced dataset is a dataset in which the number of observations belonging to one class is considerably lower than those belonging to other class/ classes. A predictive model employing conventional machine learning algorithms could be biased and inaccurate when being employed on such datasets. This is purely because machine learning algorithms are designed to improve accuracy by reducing the error in the network. Therefore, they do not consider the class distribution, class proportion, or balance of the classes in their classification process. A predictive machine learning model being bias or inaccurate can be predominant in scenarios where the minority class belongs to the malicious activities and the anomaly detection is extremely crucial. This includes scenarios such as: occasional fraudulent transactions in banks, irregular insider threats, rare disease identification, natural disaster such as earthquakes, and periodic malicious activities on critical infrastructures (e.g. infrequent attacks on nuclear power plants or water supply systems in a city). Given the importance of these scenarios, an inaccurate classification by a predictive machine learning model could cost thousands of lives or huge cost to individuals and/or organisations. There are several techniques to solve such class imbalance problems using various sampling/non-sampling mechanisms e.g. oversampling, undersealing and SMOTE as well as ensemble methods and cost-based techniques. However, the importance of an imbalanced dataset has not been clearly and adequately investigated in the literature particularly for machine learning-based solutions for insider threat detections.

Therefore, in this paper, our focus is on an extremely imbalanced dataset of insider threats where the number of events belonging to the malicious class is considerably lower than those belonging to the benign class. We use spread subsample (Weka. Class SpreadSubsample, 2018) as a popular balancing technique. The filter allows you to specify the maximum spread between the rarest class and the most common one. For example, given an imbalanced dataset, you may indicate that there should be a figure of 3:1 difference in class frequency. For this, the original dataset first fits in the memory then a random subsample of a dataset will be produced given the identified maximum spread between two classes. In this paper, we specify the maximum "spread" between the rarest class (i.e. malicious events) and the most common class (i.e. benign events) as "1" representing uniform distribution between two classes. This allows us to keep all the malicious events plus the equal number of the benign events selected randomly which results in having a uniform distribution of malicious and benign events.

In this paper, we raise the following specific research questions:

RQ1: Does balancing the dataset during the pre-processing phase improve metrics such as: Classification Accuracy (CA), Time taken to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F) in comparison with the same metrics but on an imbalanced dataset?

RQ2: What are the important parameters for each classifier that configuring them could have an impact on the classification results?

RQ3: Does changing these parameters with different values improve metrics such as: Classification Accuracy (CA), Time taken to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F) in comparison with running the classifiers with the default parameters?

Additionally, this paper provides a comprehensive explanation and investigation on the data pre-processing stage which is a crucial part of any data mining/ machine learning projects. For this, a clear step-by-step description is provided by the authors. Furthermore, we provided six comprehensive

data breach scenarios with full justifications and explanations which includes: data theft, privileged user data breach, endpoint security, shadow it risk, data security, and sensitive folders protection.

The remainder of this paper is organised as follows. In Section 2, we review the related work based on our research questions followed by our data analysis in Sections 3. This is trailed by implementation, results and analysis in Section 4, conclusion and future work in Section 5, and acknowledgment as well as references.

2. RELATED WORK

Insider threat problems have been studied widely in the existing literature. This covers extensive categories such as: traitor and masquerader, real-time and non-real time, host-based, network-based and hybrid (host-based plus network-based), user profiling, use of different datasets as well as machine learning-based solutions.

Given that this paper focuses on insider threat detection using supervised machine learning algorithms for an imbalanced dataset, in this section, we address the existing work where supervised, unsupervised, and semi- supervised approaches have been employed. For completeness, we first briefly explain four popular categories of Machine Learning (ML) techniques: supervised, unsupervised, semi-supervised and transfer learnings as follows.

Supervised ML techniques such as: Nearest Neighbour methods (e.g. k-NN), Neural Networks (NNs) and Support Vector Mechanism (SVM), build data classification model from labelled training data where for every single input (e.g. x1, x2) there is a corresponding target (e.g. y1, y2). It is a classification problem, if the targets are represented in some classes and alternatively a regression problem, if the targets are continuous. As the name suggests, there is a strong element of "supervision" in supervised learning. Lengthy training time, high training costs and the requirements for large amounts of well-balanced data are some of the drawbacks for this methodology.

In unsupervised ML techniques such as: Gaussian Mixture models, Self-Organising Map (SOM) and Graph-Based Anomaly Detection (GBAD), the data is unlabelled. This means for each single input (e.g. x1, x2), there is no target output nor reward from its environment. The goal of unsupervised learning is to find hidden structure or relation amongst unlabelled data for clustering or compression purposes. However, despite supervised learning, since there is no element of "supervision" and there is no label, unsupervised learning can't provide evaluation of the action. Additionally, neither pre-determination on the number of classes nor getting very specific about the definition of the classes is possible.

Semi-supervised learning techniques such as: Self Training, Generative Models, Graph Based Algorithms and Semi Supervised Support Vector Machines (S3VMs), fall between supervised and unsupervised learning by making use of both labelled data (typically small amount) and unlabelled data (typically large amount) for training. Therefore, the goal is to overcome one problem from each category: not having enough data in supervised learning and having no classification in un-supervised learning. Hence, given that generating labelled data is often costly as it needs a lot of resources such as manpower and computation while unlabelled data is generally not, semi-supervised learning sounds like a powerful approach particularly for big data. However, it suffers from some limitations. For instance, when the data patterns change in a semi-supervised approach, old assumptions about labelled data may screw up the new unlabelled data.

Transfer learning is a machine learning technique with an extra source of knowledge in addition to traditional training data. Some work in transfer learning includes extending popular classification and probability distribution systems such as artificial neural networks, Bayesian networks and Markov Logic Networks. Three common measures by which transfer learning might improve learning comprises: 1) performance improvement in target task with transfer learning in addition to the traditional learning from data compared to the non-transfer learning scenario, 2) total time to fully learn the target task with and without transfer learning process, and 3) initial performance in

the target task learning using only the transferred knowledge compared with the scenario where only traditional training data is used (Torrey, 2009).

However, negative transfer avoidance where applying transfer methods decreases performance, automation of task mapping to translate between task representations, transfer between more diverse tasks, and performing transfer in more complex testbeds are the challenges facing transfer learning techniques.

A summary of the recent work on insider threat detection using machine learning techniques is presented in Table 1. We classify them based on different metrics such as: ML algorithms e.g. Supervised (S) (Singh, 2014) and (Gavai, 2015), Unsupervised (U) (Parveen, 2011),)Tuor, 2017), (Böse, 2017) and (Parveen, 2012) or Semi-supervised (Se) (Gavai, 2015), and (Veeramachaneni, 2016). The addressed work in Table 1 is also classified based on the source of data e.g. user logs or available insider threat datasets, setup environment, experiment results and performance metrics e.g. False Positive (FP) and False Negative (FN) rates. The specifications related to our work in this paper is also presented in Table 1.

Authors in (Singh, 2014) employed supervised Modified k-Nearest Neighbour (k-NN) with Community Anomaly Detection System (CADS) and metaCADS on non-real-time user access logs. CADS and metaCADS are the frameworks that don't depend on any prior knowledge such as user role or access control mechanism to detect anomalies. While CADS includes two steps of pattern extraction and anomaly detection, metaCADS contains two more steps of network construction assignment and complex category interface. They claimed that the Modified k-NN beats k-NN. However, their presented result doesn't show this and are rather unclear. No performance metrics e.g. FP or FN are presented either. Additionally, some examples on user access logs such as user – subject relationship and the subject – category relationship would have made their work clear.

Authors in (Hashem, 2016) used supervised SVM on a combination of real-time human biological signal patterns such as ElectroEncephaloGraphy (EEG) and ElectroCardioGram (ECG) with ten volunteers and three scenarios that included malicious and benign activities. They captured 86% average detection accuracy with EEG which was increased by 5% after adding ECG signals. However, justification for not including ElectroMyoGraphy (EMG) signals in addition to EEG and ECG signals has not been identified. Additionally, their results from Principal Component Analysis (PCA) have not been clearly detailed. Given that their work relies on users wearing the headset and sensors all the time to measure the signals continuously, the risk of wearing them on user's health is also not clear. Additionally, not only wearing headset and sensors constantly is inconvenient for a user but an insider is less likely to wear them effectively knowing that his/her activates are going to be monitored continuously.

Authors in (Tuor, 2017) employed unsupervised Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs) on Computer Emergency Response Team (CERT) Insider Threat Dataset v6.2 (CERT) with Tenserflow. Given that their work functions on raw system logs from CERT insider threat dataset, they considered it as equivalent to raw streaming data thus calling their work online and real time. They captured Cumulative Recall (CR-k) for daily budgets of 400 and 1000 which showed that DNNs and RNNs outperform Principal Component Analysis (PCA), SVM and Isolation Forest (IF). Given that their work models only normal behaviour and uses anomaly as an indicator of potential malicious activities, FP rate has not been clearly addressed. CERT dataset is based on CERT model which provides advantages such as sufficient study on insiders' motivation and psychological or behavioural aspects of their crime or offense in the field of insider threat. However, CERT has some drawbacks. For example, there is no insider threat classification for the raw system logs in this dataset. CERT model is also complex even after the latest improvements. This means for using this model a group of experts is always needed to perform prediction.

Authors in (Parveen, 2012) used unsupervised incremental sequence learning combined with compression learning for insider threat detection. They worked on 168 real benign trace files of users in University of Calgary in addition to their 25 malicious artificial files. The trace files are the Unix C shel collected from four groups of people and categorised based on their programming skills.

Table 1. Related work on insider threat detection using machine learning techniques

Author, year	ML: S/ U/ Se	ML algorithms	Source of Data	Environment, Configurations, Framework	Experimental results	Captured performance (e.g. FP and FN)	Real Time?
(Singh, 2014)	S	Modified k-Nearest Neighbour (k-NN)	User access logs	Community Anomaly Detection System (CADS) & metaCADS frameworks	Modified kNN beats k-NN	N/A	No
(Hashem, 2016)	S	Support Vector Machine (SVM)	Ten volunteers equipped with: Emotiv EPOC headset and OpenBCI sensors	Electroencephalography (EEG) and the electrocardiogram (ECG) signals	91% average detection accuracy	Presented: Accuracy, Precision, Recall, F-measure, and Error-rate	Yes
(Parveen, 2011)	U	Ensemble- Graph-Based Anomaly Detection (E-GBAD) with Stream Mining and Graph Mining	1998 Lincoln Laboratory Intrusion Detection Dataset	System logs from the dataset each specifies by a token with eight attributes	E-GBAD approach is more effective than traditional single-model approach	0% FN and a lower FP rate than single-model	No
(Tour, 2017)	U	Deep Neural Networks (DNNs) & Recurrent Neural Networks (RNNs)	Computer Emergency Response Team (CERT) Insider Threat Dataset v6.2	Used Tenserflow. Tuned the data based on random hyper-parameter search e.g. learning rate of 0.01 for both DNN and RNN	DNNs and RNNs outperform Principal Component Analysis (PCA), SVM and Isolation Forest (IF)	Presented: Cumulative Recall (CR-k) for daily budgets of 400 and 1000	Yes
(Parveen,, 2012)	U	Compression and incremental learning	User trace files of the Unix C shel in University of Calgary in addition to their own files	168 real benign files collected from four groups of people in addition to their 25 artificial malicious files	Their approach performed well with limited number of FP compared to static approach	Presented: Average FP and average TP rates	No
(Bose, 2017)	U	Unsupervised k-NN and k dimensional (k-d) tree	Generated by DARPA ADAMS program	Real- time Anomaly Detection In Streaming Heterogeneity (RADISH) system	k-d tree is much faster than kNN. RADISH performance and its accuracy presented	N/A	Yes
(Veeramachaneni, 2016)	Se	Matrix Decomposition, Neural Networks, and Joint Probability	A three-month real log data produced total of 3.6 billion lines generated by enterprise platform	Components: an analytics platform, an unsupervised rare event modeller, a feedback platform, and a supervised learning module	The system learns to defend against unseen attacks	Presented: TP and FP rates; TP rate improved; FP rates reduced when time progresses	Nearly real time
(Gavai 2015)	Se	Unsupervised Modified Isolated Forest + Supervised Random Forests	A real-world dataset named 'Vegas' for benign activities in addition to artificially injected insider threat events	An unsupervised and a supervised approach: to identify abnormality and to label quitters, respectively	Unsupervised: ROC score of 0.77%, supervised: accuracy of 73.4%	Presented: ROC, Recall and Precision	No
This work, 2019	S	Supervised machine learning algorithms: J48 SVM, Naïve Byes (NB), and Random Forest (RF)	Six demo scenarios obtained from (ZoneFox, 2017)	Captured eight features for five users in four consecutive days for hours to construct five user profiles	Balancing the dataset doesn't improve performance metrics; the impact of using different parameters for classifiers is significantly stronger on imbalanced dataset	Presented: Accuracy, Time taken to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F)	No

They captured FP and TP average rates that shows their work performed well compared to the static approach. However, the malicious anomalous data is artificial and crafted by the author. Additionally, this public dataset (Greenberg, 1988) is now dated.

Authors in (Veeramachaneni, 2016) proposed a nearly real-time supervised and unsupervised system with an analyst-in-the-loop against unseen attack. The unsupervised part of their system learns a model that can identify extreme and rare events in data. The rare events then presented to the system analyst who labels them as either malicious or benign. At the end, the labelled data is provided to the supervised learning part of their system which produces a model that can predict whether there will be attacks in the following days. On the unsupervised part of their model, they employed Matrix Decomposition, Neural Networks and Joint Probability to detect rare events each of which generates a score indicating how far a certain event is from others. They used a total of 3.6 billion real log lines produced within 3 months and presented that the TP rate improves by almost three times and FP rate reduces by five times as time progresses.

Authors in (Gavai, 2015) employed unsupervised Modified Isolation Forest (IF) and supervised Random Forest algorithms to discover insider threats based on employees' social and online activity data. Their approach includes generating some features based on email content, work-practice and online activity which is then used in the unsupervised part of the proposal to identify suspicious behaviour. These features are also used in the supervised part of their proposal in conjunction with quitting labels to develop a classifier. They used a real-world dataset named Vegas for benign activities in addition to their own artificially injected insider threat events. The events include patterns for email communication, web browsing, email frequency, and file and machine access. They obtained a ROC score of 0.77% for the unsupervised part of their work and a classification accuracy of 73.4% for the supervised part showing that their approach is successful in detecting insider threat activity as well as quitting activity. However, the malicious anomalous data is artificial and crafted by the authors.

The work presented in this paper differs from the related works described above as we focus on employing machine learning techniques on extremely imbalanced dataset in which the malicious class is in the minority and the benign class is in the majority (i.e. the malicious events are less than 1.3% of the total dataset). Given that the importance of the class imbalance problem has not been addressed in the existing literature for the insider threat, our methodology for exploring the impact of data balancing derived from (Galar, 2011), (Hanifah, 2015) & (Pavlov, 2010) where different balancing techniques have been discussed including undersampling, oversampling and hybrid-based approaches from which we employ a popular undersampling technique called spread subsample. Our work in this paper is built upon our previous work (Moradpoor, 2017) where we used Self-Organising Map (SOM) in conjunction with Principal Component Analysis (PCA) for insider threat detection on the same imbalanced dataset that we use in this paper. However, in our previous work, the dataset was unlabelled. Given that it is less likely to have a balanced dataset in a real-world scenario, knowing how to deal with an imbalanced dataset is not only crucial but also more realistic.

For the classifiers, we consider J48 decision tree, Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF) algorithms and for performance metrics we study Classification Accuracy (CA), Time taken to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F). These performance metrics are mainly derived from (Buczak, 2015) where a comprehensive survey of data mining and machine learning techniques for cyber security intrusion detection has been fully discussed.

Additionally, we provide a comprehensive review and explanations on the data pre-possessing phase. This phase, which is missing from the existing work on ML-based insider threat detection e.g. from (Singh, 2014), (Parveen, 2011), (Tuor, 2017), (Bose, 2017), (Hashem, 2016), (Gavai, 2015), (Parvenn, 2012) and (Veeramachaneni, 2016), is an extremely important part of any machine learning projects given that it transfers a raw or original dataset to an understandable and meaningful format. This part of our work is encouraged by (Hamed, 2018) where a taxonomy of data pre-processing techniques for intrusion detection systems has been discussed. In this paper, the data pre-processing

phase includes outlier identification from data cleaning stage, data decomposition and data conversion from data transformation stage, and data balancing from data reduction stage. The comprehensive explanations on data pre-processing phase on extremely imbalanced dataset in this paper can be used as a guide and can be employed on any data mining/machine learning projects.

Furthermore, despite (Singh, 2014), (Parveen, 2011), (Tuor, 2017), (Bose, 2017), (Hashem, 2016), (Gavai, 2015), (Parvenn, 2012) and (Veeramachaneni, 2016), this paper contains six comprehensive demo setups which covers all the serious data breach scenarios: data theft, privileged user data breach, endpoint security breach, shadow it risk, data security, and sensitive folders breach. Besides, to make the scenarios more realistic, it includes different arrangements for three groups of staff within an organisation: permanent staff (x3 scenarios), temporary staff (x1 scenario) and third-party staff (x2 scenarios). Our methodology for defining, justifying, and developing the six scenarios above is enthused by (Lindauer, 2014) where generating test data for insider threat detectors has been described. The work in (Legg, 2015) also inspired our work in terms of using user and role-based profiles for automated insider threat detection.

3. ANALYSING THE DATA

In this paper, six demo scenarios have been identified which help to shape the dataset used in our experiments. They have been developed by (ZoneFox, 2017), which is a market leader in User Behaviour Analytics, to ensure familiarisation of the insider threats in an organisation. In this section, we discuss: the six demo scenarios, the original dataset that we used for our experiments and the data pre-processing stage (outlier identification, data decomposition and data conversion).

3.1 DEMO Scenarios

Our six demo scenarios include different setups for three groups of staff: permanent, temporary and third party. This contains three scenarios for permanent staff: Data Theft, Endpoint Security Processing and Shadow IT Risk, one scenario for temporary staff: Privileged User Data Breach and two scenarios for third-party member of staff: Data Security and Protect Sensitive Folders. To clarify a permanent staff is a long-term member of staff who works for instance in company's engineering or sales team e.g. Charlotte, Rebecca and Laura. A temporary staff is a short-term member of staff who has been employed for the busy period e.g. Timmy. A third-party member of staff is an individual who has been employed to work with one of the client systems e.g. Colin. The names are all fictional and have been used for the sake of making our demo scenarios clear. All six scenarios are presented in Table 2 and defined as follows.

3.1.1 Demo Sceanrio 1: Data Theft

A Data Theft for an organisation commonly includes stealing sensitive information related to its staff, clients or business e.g. usernames, passwords, credit card information, medical records or business secrets. In Demo Scenario 1, we focus on the data theft threat from employees within a given organisation by following the Insider Threat Kill Chain (ITKC). ITKC identifies human resources as the greatest risk to organisations and discusses how members of staff within organisations can work together to help avert these risks before they become a problem. To achieve this, we consider four groups of people within the organisation: 1) people who intended to leave their job and already handed in their notice, 2) people who look around file servers repeatedly, 3) people who download backup software on their desktop computers and 4) people who copy zip files to their removable devices. For example, as it is shown in Table 2 Demo Scenario 1, Charlotte, who is a permanent member of staff working in the company's engineering team, backs up files to a removable disk drive.

3.1.2 Demo Scenario 2: Privileged User Data Breach

Intruders need privileged access, for instance admin's username and password, to carry out malicious activities and proceed attacks to the next phase. Mischievous activities such as installing malicious software, e.g. malware, ransomware or backdoor, stealing sensitive information, e.g. usernames and passwords, or even disabling hardware and/or vital software on a victim's computer, e.g. anti-malware or anti-spyware.

That is why privileged user accounts are in such high demands by intruders. In fact, there can be so many shared privileged accounts within an organisation with no management knowledge on where all those accounts reside or who has access to them. In Verizon Data Breach Investigations Report for 2016 (Enterprise, 2017), 63% of all breaches were due to using weak or common passwords while 53% of them were down to the misuse of privileged accounts. Privileged accounts include: local admin accounts, privileged user accounts, domain admin accounts, emergency accounts, service accounts and application accounts. An efficient privilege management system would automatically identify these accounts and bring them under a management system umbrella. In Demo Scenario 2, pinpointing privileged user data breach is done by going through the following steps: 1) identifying privileged user accounts, 2) identifying sensitive files that could only be accessed using those accounts, and 3) monitoring the access to those sensitive files. For example, as it is shown in Table 2 Demo Scenario 2, Timmy, who is a temporary member of staff who has access to privileged user accounts, accesses a file, i.e. minutes of company's February meeting, on the network file system that he does not need access to.

3.1.3 Demo Scenario 3: Endpoint Security Processing

In an organisation, endpoints include desktop computers, mobile devices (e.g. laptops, smartphones and tablets), printers, scanners or even bar code readers. Endpoint security management is a network security approach that requires endpoint devices to follow the security policy of a given organisation. This needs to be met before as well as after granting access to network resources. In Demo Scenario 3, we consider endpoint security processing by monitoring the staff activities on desktop computers across the organisation. For instance, turning off or disabling a system's anti-malware e.g. popup

Table 2. Six demo scenarios of users

Demo scenarios	Staff type	Staff name	Application	Resource	Description
Demo scenario 1: *Data Theft*	Permanent	Charlotte	bbackup.exe	rm:\\e:\\myusb\\backup.zip	Charlotte backs up files to a removable disk drive.
Demo scenario 2: *Privileged User Data Breach*	Temporary	Timmy	N/A	nfs:\\.....\\fileshare2\\ boardminutes\\minutes - february.txt	Timmy accesses a file on the network file system that he does not need to access to.
Demo scenario 3: *Endpoint Security Processing*	Permanent	Laura	Savservice.exe (av software)	N/A	Laura deactivates the anti-virus software on her computer.
Demo scenario 4: *Shadow IT Risk*	Permanent	Rebecca	dropbox.exe Skype.exe	c:\\users\\rebecca\\dropbox\\ plan2.doc c:\\users\\rebecca\\dropbox\\ plan1.doc nfs:\\....\fileshare1\\ engineeringplans\\plan1. doc nfs:\\....\fileshare1\\ engineeringplans\\plan2.doc	Rebecca uses Dropbox to perform unauthorised backups to the Cloud. Rebecca uses Skype to perform unapproved uploads from network file system to Cloud.
Demo scenario 5: *Data Security*	Third-party	Colin	N/A	nfs:\\.....\fileshare1\\ engineeringplans\\plan1.doc rm:\\f:\\copyto\\remdrive\\ ipdata.txt	Colin accesses a file on the network file system that he is not supposed to access and copies it to a removable disk drive.
Demo scenario 6: *Protect Sensitive Folders*	Third-party	Colin	N/A	nfs:\....\fileshare1\PatientData\ AliceBrownMedicalRecord.txt	Colin accesses sensitive information (Alice Brown's medical record) on the network file system that he should have no need to access to.

blockers, anti-spyware, anti-spam, host-based firewalls and anti-viruses. For example, as it is shown in Table 2 Demo Scenario 3, Laura, who is a permanent member of staff working in the company's sales team, deactivates the anti-virus on her desktop computer.

3.1.4 Demo Scenario 4: Shadow it Risk

Shadow IT refers to software or applications purchased, downloaded, installed, used or managed outside or without the knowledge of an organisation's IT department. They can be defined as the IT assets that are invisible to an organisation's IT department. Shadow IT has grown exponentially in recent years. This is down to the good quality of applications in the Cloud, mobile technology growth and the rapid development in Software as a Service (SaaS), such as: Dropbox, Cisco WebEx, Google Apps, Salesforce, Skype, and Microsoft Office 365. A SaaS, which is also known as Cloud software, on-demand software or hosted software, is a way of delivering applications over the Internet. A given SaaS may or may not offer strong security protections e.g. identity management, authentication, access control, secure backup practices, data masking or data encryption. Therefore, it can expose an organisation and/or its affiliates to data loss risk and security-related threats such as insider threats. In Demo Scenario 4, we consider Shadow IT risk, which is imposed by employees within the organisation using unknown/unauthorised SaaS e.g. Dropbox or Skype. For example, as it is shown in Table 2 Demo Scenario 4, Rebecca, who is a permanent member of staff working in the company's engineering team, installs Dropbox to perform unauthorised backups and Skype to accomplish unapproved uploads to the Cloud.

3.1.5 Demo Scenario 5: Data Security

Data Security refers to protective measures applied to prevent unauthorised access and corruption to resources such as endpoint devices (e.g. computers, printers and tablets), databases, websites, and computer files. Backups, data encryption, authentication and data masking are the commonly used data security techniques. For example, data masking is a method of protecting the actual data by creating a structurally similar but fake version of an organisation's data for purposes such as training or software/algorithm testing. In Demo Scenario 5, we consider monitoring resources that a third-party member of staff is not supposed to access or make a copy from e.g. accessing a sensitive document or copying intellectual property files to a removable disk. For example, as it is shown in Table 2 Demo Scenario 5, Colin, who is a third-party member of staff, accesses and copies intellectual property data to a removable disk drive that he is not supposed to do.

3.1.6 Demo Scenario 6: Protect Sensitive Folders

In general, data can be categorised as either public, restricted or private. The public data such as: press releases, course information or research publications can be available to anyone with little or no controls to protect their confidentiality. The restricted data such as: credit card numbers, passwords or personal medical information are those for which the unauthorized disclosure, alteration or destruction of them could cause a significant level of risk to an organisation or its affiliates. The private data such as: home address, birth date, gender, religious or sexual orientation are those for which the unauthorized disclosure, alteration or destruction of them could result in a moderate level of risk to an organisation or its affiliates. In Demo Scenario 6, we focus on protecting folders with restricted data e.g. folders that carry medical records for individuals maintained by the organisation. For example, as it is shown in Table 2 Demo Scenario 6, Colin, who is a third-party member of staff (i.e. a contractor), accesses restricted data (i.e. one of the staff's medical records) which he should have no need to access.

3.2 Original Dataset

In this section, we explain the dataset that we used for our experiments. As mentioned before, the original dataset has been given by (ZoneFox, 2017) and covers all the six scenarios that we defined

in the previous section. It includes Charlotte, Rebecca, Laura, Timmy and Colin's user profiles captured on four consecutive days. The dataset also contains the administrator's network activities. As discussed before, each user is either a permanent member of staff (e.g. Charlotte, Rebecca, Laura or Administrator), a temporary member of staff (e.g. Timmy) or a third-party member of staff (e.g. Colin). The original dataset is in .CSV format and contains 2643 lines of raw data which includes six features: Date-Time, machine_ID, user_ID, application, action and resource. Each line of the dataset identifies an action done by one of the users in the user pool (i.e. Charlotte, Rebecca, Laura, Timmy, Colin or Administrator). For example, one line of the dataset specifies that on 2016-02-23 at 16:30:27 (Date-Time), employee A (user_ID) used computer B (machine_ID) to run backup.exe (application) and write (action) the backup into path D (resource).

3.3 Data Pre-Processing

In this section, we explain the pre-processing phase employed on our original dataset given by (ZoneFox, 2017). This phase is an important part of any machine learning projects given that it transfers a raw or original dataset to an understandable and meaningful format. Raw data is often incomplete, noisy, inconsistence and/or lacking certain behaviours, attributes or styles and is likely to generate or comprise errors if fed intact into machine learning algorithms. Data pre-processing is a proven method of resolving these issues. It comprises stages such as: data cleaning, data integration, data transformation, data reduction and data discretisation each containing various tasks. For example, data transformation comprises tasks such as normalisation and aggregation while data cleaning includes filling in missing data, outlier identification, outlier removal and inconsistency resolution.

In this paper, the data pre-processing phase includes four tasks of: outlier identification from data cleaning stage, data decomposition and data conversion from data transformation stage, and data balancing from data reduction stage as follows.

3.4 Data Cleaning

In any data science project, data cleaning is a critical part of the data pre-processing phase. This includes tasks such as: filling in missing values, identifying outliers, smoothing out noisy data or correcting inconsistent data. For example, for filling in missing values a learning algorithm such as Bayes or decision tree can be used to predict them. Additionally, domain knowledge or expert decision can be employed to correct inconsistent data. In this paper, we used outlier identification task from data cleaning stage as follows.

3.4.1 Outlier Identification

In data science, "outliers" are values that "lie outside" the other values e.g. in the scores of: {3,22,25,23,29,33,85}, 3 and 85 are outlier given that they are far away from the main group of data: {22,25,23,29,33}. In data mining, outlier identification, which is also known as anomaly detection, refers to identification of items, events or behaviours which do not follow an expected pattern. It is an observation of the data that deviates from other observations so much that it awakens suspicions that it was generated by a different and/or unusual mechanism (Williams, 2002). In the data pre-processing phase, outlier identification is a part of the data cleaning stage and includes tasks such as binning, clustering and regression. In this paper, outlier means insider threat and outlier identification refers to detection of digital tasks performed by employees which they are not supposed to do/or they should have no need to do. In order to perform outlier identification on our original dataset, we surfed through each event manually and marked it as either 1, representing an outlier event, or 0, representing a non-outlier event. This is decided by considering the six demo scenarios explained in the previous section (i.e. Data Theft, Endpoint Security Processing, Shadow IT Risk, Privileged User Data Breach, Data Security, Sensitive Folders Protection) along with the individual's role within the organisation. The original dataset is in .CSV format and includes 2643 lines of user activities for five individuals: (i.e. Charlotte, Rebecca, Laura, Timmy and Colin). The outlier identification task

resulted in identifying a total of 33 outlier activities out of 2643 events within the original dataset. As it is depicted in Table 3, total outlier events of 6, 5, 11, 1 and 10 belong to Charlotte, Rebecca, Colin, Laura and Timmy, all respectively.

Given that this paper is based on a supervised machine learning approach, the outlier identification task adds an extra feature of "Outlier" to our dataset which is also known as "label". Outlier identification task or labelling the entire dataset is done to provide a platform to evaluate the performance of our supervised machine learning approach. It also assists with our future work which is based on an unsupervised machine learning approach. Additionally, labelling the entire dataset will assist us in implementing a semi-supervised approach for our possible future work which is an implementation of a supervised method in addition to the unsupervised method that could have a potential to improve the accuracy rate for insider threat detection.

3.5 Data Transformation

Machine learning algorithms were unlikely to process any data correctly and/or produce any accurate results such as predictions if the data does not follow a similar type. Therefore, we employed data transformation stage on our original dataset during the pre-processing phase to transform them into a similar data type. This includes two steps of data decomposition and data conversion as follow.

3.5.1 Data Decomposition

In a dataset, decomposing some values into multiple parts will help a machine learning algorithm in capturing more specific relationships. For instance, data decomposition on a feature such as "date" represented as "Tues; 01.04.2017" into: day of a week and month of a year may provide more relevant information. In fact, data decomposition is the opposite of data reduction given that it will add more data to the original dataset.

In this paper, we employ data decomposition on Date-Time, application, action and resource features in our dataset to capture more specific relationship between the individual user behaviour and insider threat.

For instance, in our original dataset, Date-Time represented as 2016-02-23T16:26:33Z indicating a user event that has happened on 23th of February 2016 at 16:26:33 hours. This includes two characters: T, which can be read as an abbreviation for Time, and Z, which stands for zero-time zone as it is offset by 0 from the Coordinated Universal Time (UTC).

After data decomposition Date-Time feature breakdown to: Year (2016), Month (02), Season (Winter), Day (23), Week Day (Tuesday), Week Portion (Middle of week), Hours (16), Minutes (26), Seconds (33) and Time of Day (Afternoon). Likewise, the application feature is decomposed to application type (i.e. system application or user application) and application ID, action feature is decomposed to action type (i.e. related to processes or related to files) and action ID and resource feature is decomposed to resource location (i.e. local computer, local network, removable memory or Cloud), file extension (e.g. .txt, .zip or. bkc) and folder depth (e.g. 1, 2 or 3).

Table 3. Outlier identification

Employee Name	Total Events 441/2643454??	Outlier Events 33/2643
Charlotte	146	6
Rebecca	75	5
Colin	57	11
Laura	135	1
Timmy	28	10

3.5.2 Data Conversion

Often, machine learning algorithms require the data in specifics ways before feeding them into the model. This can be done during the data conversion task which is the conversion of data from one format to another format e.g. from categorical data to numeric values. Categorical data are the data that contain label values, which is often limited to a fixed set, rather than numerical values. Some examples include: "week day" variable with the values of: "Monday", "Tuesday" and "Wednesday" or "season" variable with the values of: "spring", "summer" and "autumn". Therefore, given that most machine learning algorithms can't operate on categorical data directly and require all input and output variables to be numeric, integer encoding also known as numeric conversion and one-hot encoding can be used. In integer encoding, each categorical data is assigned an integer value e.g. "Monday" is 1, "Tuesday" is 2, and "Wednesday" is 3. However, in one-hot encoding, a binary variable is added for each value e.g. in the "week day" variable example above "Monday" is 001, "Tuesday" is 010, and "Wednesday" is 011.

In this paper, the data conversion task has been employed on 13 out of 20 features of our dataset. This includes numeric conversion for: Date-Time, season, week day, week portion, time of day, machine_ID, user_ID, application_type, application_ID, action_type, action_ID, resource_location and file extension.

For example, in our main dataset, the Date-Time feature for each event contains standard date and time format including both numeric and text character. For instance, 2016-02-23T16:26:33Z represents a user event that happened on 23rd of February 2016 at 16:26:33 hours. This includes two characters: T for Time, and Z for zero-time zone as it is offset by 0 from the Coordinated Universal Time (UTC). For Date-Time conversion, we first split this feature to date and time. This gives us: 2016-02-23T and 16:26:33Z for our running example above. We then removed T and Z characters and formatted the date to: 23/02/2016 while time stays the same: 16:26:33. We then converted date and time to a UNIX timestamp which results in 1456185600 and 59193 for the date and time, respectively. In the last step, we combined them together which gives us: 1456244793. UNIX timestamp is the number of seconds between a particular date and the Unix Epoch on January 1st, 1970 at UTC. We run the Unix timestamp conversion on the Date-Time feature for the entire dataset.

Besides this, as we discussed in the previous section, the Date-Time feature was further split into year, month, season, day, week day and week portion (for Date feature) and hours, minutes, seconds and time of day (for Time feature) during the data decomposition task. Therefore, we allocate a 1-4 range to season, 1-7 to weekday, 1-4 to week portion and 1-4 to time of day in the data conversion task. However, year, month, day, hours, minutes and seconds remain unchanged during this task.

Furthermore, in our original dataset, each machine_ID is a combination of numbers, uppercase and lowercase letters e.g. 4RcZBZz. We defined 15,000 – 19,000 range for machine_ID data conversion from which an integer value is assigned to each computer. For example, 16002 has been assigned to a computer with an id of 4RcZBZz. Likewise, 1000 – 1500 range is allocated to the user_IDs in our dataset. This means a unique numerical value for Administrator, Charlotte, Rebecca, Laura, Timmy and Colin e.g. 1301 has been assigned as a user_ID to Colin.

Similarly, 20 – 99 range has been assigned to application_ID, 200 - 499 to action_ID, 0-4 to resource_location and 0-19 to file extension. Moreover, in terms of application_type and action_type, 0 represents systems application and process related actions and 1 represents user application and file related actions, both respectively.

To put it in a nutshell, seven features of year, month, day, hours, minutes, seconds and folder depth, remain unchanged during the data conversion task. All the arrangements for data conversion have been identified in Table 4.

3.6 Data Reduction

This stage of the data pre-processing phase includes steps as follows (Data pre-processing, 2018). "Reducing the number of attributes": for instance, removing irrelevant attributes or using principle

component analysis to search for a lower dimensional space that can best represent the data. "Reducing the number of attribute values": for example, by clustering where similar values grouped in clusters. "Reducing the number of data samples": for instance, within the majority class to reduce the degree of imbalance data distribution and produce a balanced dataset. This is also known as data balancing. In this paper, we only employ data balancing during the data reduction stage as follows.

3.6.1 Data Balancing

In machine learning and data mining, the imbalanced class distribution is a scenario where the number of instances belonging to one class is significantly lower than those that belong to the other classes. In such a scenario, as we have in this paper, the classification could be biased and inaccurate. This happens due to the fact that machine learning algorithms are usually designed to increase the accuracy by reducing the error. Therefore, to achieve this, they do not consider the class distribution, class proportion, or balance of classes. There are various approaches for solving class imbalance problems. This includes two areas of: 1) resampling and 2) classifier modifications. In this paper, we focus on the first category where we reassemble the original dataset to provide two balanced classes (i.e. malicious and benign). For this, we use Weka's Spread Subsample feature (Weka. Class SpreadSubsample, 2018) where the original dataset must first fit entirely in the memory. This includes the entire imbalanced data which contains malicious and benign events. Then we need to specify the maximum "spread" between the rarest class (i.e. "malicious" which is labelled as class "1" with the total events of "33") and the most common class (i.e. "benign" which is labelled as class "0" with the total events of "2643"). For instance, "0" for distributionSpread of Spread Subsample means no maximum spread, "1" means uniform distribution and "10" means allow at most a 10:1 ratio between the classes. In this paper, we chose "1" representing uniform distribution between two classes. This allows us to keep all the malicious events (i.e. 33 malicious events) plus the equal number of the benign events selected randomly (i.e. 33 benign events). At the end, we will have 66 events in total with a uniform distribution of malicious and benign events (i.e. 33 malicious and 33 benign events).

4. IMPLEMENTATION, RESULTS AND ANALYSIS

In this section, we explain our captured results after running a number of popular machine learning algorithms with different parameters on our datasets. This is done on two copies of our datasets: balanced and imbalanced. Both datasets have been passed through the data pre-processing phase including: outlier identification, data decomposition and data conversion. However, as the name says, the balanced dataset includes an extra step of data balancing. We use Weka3 (Weka, 2018) in our experiments which is a popular and powerful tool for data mining and machine learning. It has a collection of machine learning algorithms and contains tools such as data pre-processing, classification and visualization. Weka is also well-suited for developing new machine learning schemes. The results captured from our machine learning schemes are analysed as follows.

4.1 Supervised Machine Learning Results

Supervised learning can be categorised into classification problems when the output is a class and regression problem when the output is a real value. Some popular supervised machine learning algorithms are: Naive Byes (NB) for regression and classification problems, Linear Regression (LR) for regression problems, Random Forest (RF) for regression and classification problems, Support Vector Machines (SVM) for classification problems, and Neural Networks (NNs) for regression and classification problems. Given our datasets from the earlier section, the problem in this paper is a classification problem as we want our supervised approach to explicitly identify a given event either as benign (i.e. it belongs to class "0") or malicious (i.e. it belongs to class "1"). In this section, we employ a number of popular machine learning algorithms for instance J48 decision tree, Support Vector

Table 4. Data conversion

Field name		Raneg						
Date-Time	UNIX timestamp conversion							
Date	year	unchanged; e.g. 2016						
	month	unchanged; e.g. 2 (i.e. February)						
	season	spring	summer	automn	winter			
		1	2	3	4			
	day	unchanged; e.g. 23						
	week day	numaric values: 1,2,3,4,…,7 (e.g. 1 for Mon and 7 for Sun)						
	week portion	beggining of week	middle of week	end of week	weekend			
		1 (i.e. for Mon)	2 (i.e. for Tues - Wed)	3 (i.e. for Thur -Fri)	4 (i.e. Sat - Sun)			
Time	hours	unchanged						
	minutes	unchanged						
	seconds	unchanged						
	time of day	morning	afternoon	evening	night			
		1	2	3	4			
machine_ID		15,000 – 19,000						
user_ID		1,000 – 1,500						
application	type	0: System applications		1: User applications				
	ID	20 – 49; e.g. 43 for taskmgr.exe		50 – 99; e.g. 73 for backup4all.exe				
action	type	0: Related to processes		1: Related to files				
	ID	200-399		400-499				
resource	location	N/A	local drive	network drive	removable drive	Cloud		
		0	1	2	3	4		
	file extension	N/A	.ctm	.rls	.Ing	.txt	….	.bkc
		0	1	2	3	4	….	19
	folder depth	unchanged; e.g. 2 for c:\backup\backup.ctm						

Machine (SVM), Naïve Byes (NB), and Random Forests (RF) configured with different parameters along with several techniques (e.g. single classifiers and balanced vs. imbalanced datasets).

It is worth mentioning that, for both balanced and imbalanced datasets, we ran each experiment 10 times with seed value of 1 to 10 and split percentages of 90% for training and 10% for testing.

4.1.1 Result Comparisons

There are several parameters available in the J48 classifier (Weka. Class J48, 2018). However, we considered its two important parameters: Confidence Factor, also known as C, and Minimum Number of Objects, also known as M. The C is used for pruning where additional steps are added to look at what nodes or branches can be removed to make the tree smaller and easier to understand without

affecting the performance too much. In general, smaller values for C incur more pruning. The M identifies the minimum number of instances or events per leaf. The default and configured values (also known as hyperparameters) for (C, M) are (0.25, 2) and (0.5, 10), all respectively.

There are several parameters available in the SVM classifier (Weka. Class SMO, 2018). However, we considered its important parameter of: Complexity which is also known as C (Saarikoski, 2011). The C value tells the SVM optimisation how much we want to avoid misclassifying in each training example. Generally, smaller values of C generate more misclassified examples. The default value for C is 1 which we configured to 100 in our experiments.

There are several parameters available in the NB classifier (Weka. Class NaiveBayes, 2018). However, we considered its two important parameters of: Kernel Estimator (KE) and Supervised Discretisation (SD). The KE parameter uses kernel estimator for numeric attributes rather than a normal distribution. The SD parameter uses supervised discretization to convert numeric attributes to nominal ones. By default, the KE and SD values are set to false and we changed them to true at the time to monitor their impact on the results.

Addressing the RQ1 and RQ3, in the first set of our supervised learning experiments, we want to identify the impact of: 1) balancing the dataset and 2) changing the C and M values for J48, C value for SVM, and KE and SD values for NB on the metrics such as: classification accuracy, time taken to build the model, time taken to test the model, True Positive (TP) rate, False Positive (FP) rate, precision, recall, and f-measure.

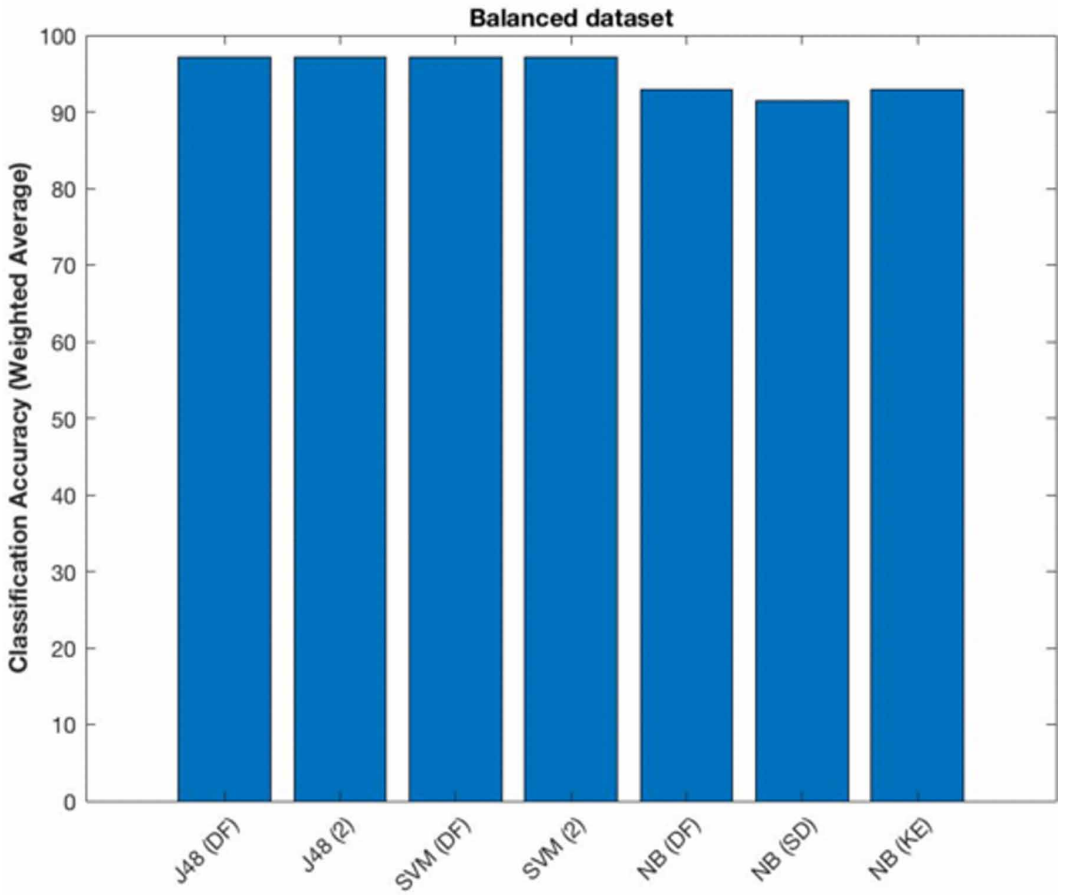
We ran each experiment ten times with the seed value of 1 to 10, measured the weighted average, and then represented it as our result. For each experiment, we considered 90% split for training and 10% for testing. This has been done for both balanced and imbalanced dataset. Additionally, J48 (DF) represents J48 with the default value of 0.25 for C and 2 for M, J48 (2) represents J48 with the configured value of 0.5 for C and 10 for M, SVM (DF) represents SVM with the default value of 1 for C, SVM (2) represents SVM with the configured value of 100 for C, NB (DF) represents NB with the default values where SD and KE are both set to false, NB (SD) represents NB with the configured value of true for SD, and NB (KE) represents NB with the configured value of true for KE.

The classification accuracies (weighted average) on the balanced and imbalanced datasets are captured in Figure 1 and Figure 2, respectively. Addressing the presented results, all algorithms except NB (DF) show a better performance on the imbalanced dataset in comparison with the balanced dataset. NB (DF) algorithm shows around 1.03% classification accuracy boost when using the balanced dataset. Therefore, answering RQ1, balancing the dataset doesn't improve the classification accuracy overall. Additionally, answering RQ3, changing parameters for each classification algorithm have more effect when using imbalanced dataset in comparison with the balanced one. In details and using imbalanced dataset, J48 (2) performs better than J48 (DF), SVM (DF) performs better than SVM (2), and NB (SD) performs best in comparison with NB (DF) and NB (KE). However, it only has effect on the balanced dataset when we use NB algorithms.

We present the time taken to build and the time taken to test the model on the balanced and imbalanced datasets in Figure 3 and Figure 4, respectively. Addressing the captured results, for all the algorithms on both datasets except NB (KE), the time taken to build the module is more and sometimes significantly more than the time taken to test the model. Answering RQ1, balancing the dataset does significantly improve the time taken to build the model for all seven algorithms. This is the same case for the time taken to test the model except for J48 (DF) in which we have an equal time for testing on both datasets. Additionally, answering RQ3, changing parameters for each classification algorithm have effect on datasets but it is rather unsteady when we compare them. In detail, on the imbalanced dataset, J48 (DF) performs better that J48 (2) and SVM (DF) performs better than SVM (2) in terms of time taken to build and to test the model. However, this is the opposite for the balanced dataset.

In Figure 5 and Figure 6, we present the true positive and the false positive rates on the balanced and imbalanced datasets, respectively. Addressing the captured results, for all the algorithms and on both datasets, the true positive rate is significantly more than the false positive rate. For the balanced

Figure 1. Classification accuracy (weighted average) using the balanced dataset



dataset, the highest true positive rate equally belongs to J48 (DF), J48 (2), SVM (DF), and SVM (2) while the lowest rate belongs to NB (SD). We also realised a nearly 2.27% jump in the false positive rate when we changed the classifier to NB algorithms. For the imbalanced dataset, the highest true positive rate belongs to SVM (DF) while all the NBs show poor performances in general. However, all the NBs perform better in terms of false positive rate in comparison with SVM and J48. Answering the RQ1, balancing the dataset decreases the true positive rate except for NB (DF) and improves the false positive rate except for NB algorithms. Additionally, answering the RQ3, changing the algorithm's parameters improve the true positive and false positive rates but with stronger impact on the imbalanced dataset than the balanced dataset.

Figure 7 and Figure 8 represent the precision, recall, and f-measure for all seven algorithms on the balanced and imbalanced datasets, respectively. In general, these three metrics are higher on the imbalanced dataset in comparison with the balanced dataset, except for NB (DF) recall. Therefore, answering the RQ1, balancing the dataset does not improve the precision, recall and f-measure overall. Given the imbalanced dataset, we noticed that changing the parameters to J48 (2), SVM (DF) and NB (SD) do improve the precision, recall, and f-measure. However, this is the case only for NB (DF) and NB (KE) on the balanced dataset. Therefore, answering the RQ3, changing the algorithm's parameters improve the precision, recall, and f-measure but with stronger impact on the imbalanced dataset than the balanced dataset.

Figure 2. Classification accuracy (weighted average) using the imbalanced dataset

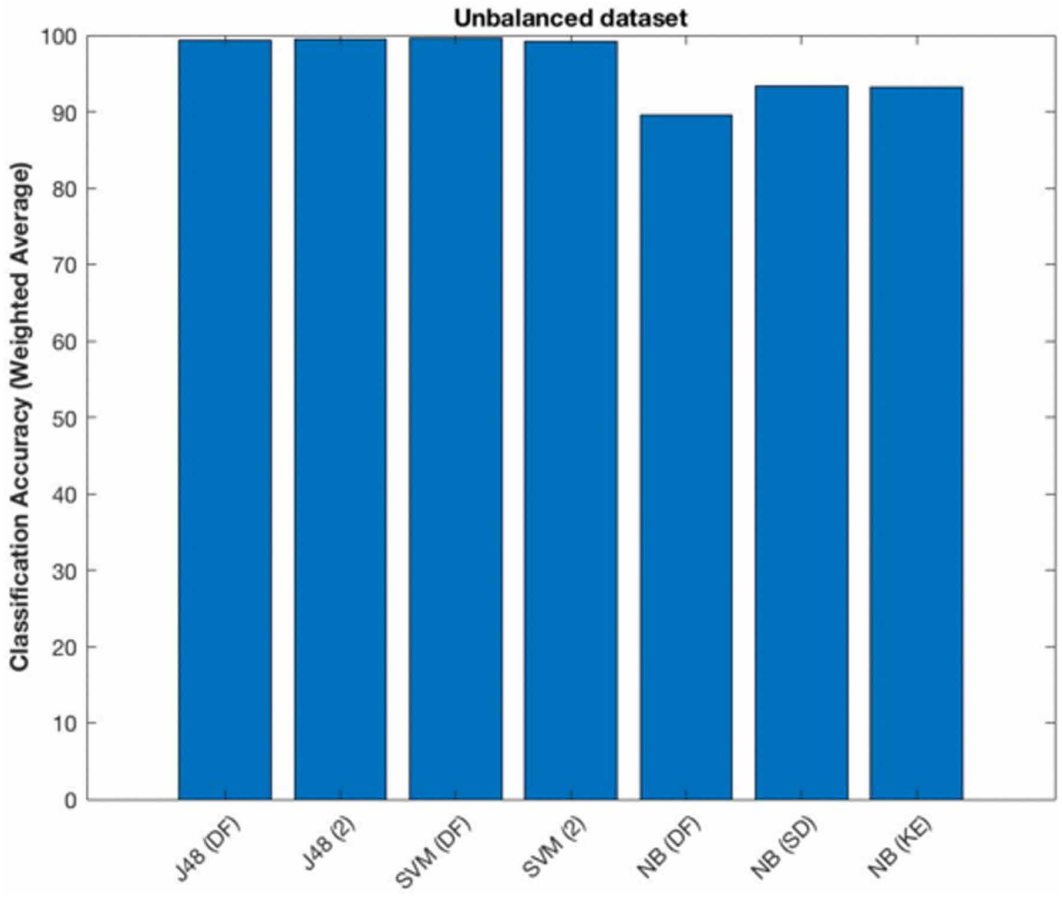


Table 5 presents the summary of our results relating to Accuracy, Time to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F) with a focus on balanced and imbalanced datasets. In this table, we identify algorithms which produce the best results with "*" in terms of the metrics above where "B" represents "Balanced Dataset" and "U" represents "Imbalanced Dataset". We score each and calculate the overall score for the table. Overall, our experiments on the imbalanced dataset (i.e. original dataset) show a better performance in comparison with the balanced one except for the time taken to build and the time taken to test the model. Therefore, answering RQ1, for each classifier, balancing the dataset doesn't improve metrics such as: Classification Accuracy, True Positive rate, False Positive rate, Precision, Recall, and F-measure in general. However, it improves the time taken to build and the time taken to test the model.

Table 6 and 7 represent the summary of our results regarding: Classification Accuracy, TB, TT, TP rate, FP rate, P, R, and F with a focus on the impact of employing different parameters in each classifier on the balanced and imbalanced datasets, respectively. In Table 6 and Table 7, "e" represents "Equal Results" and "*" represents "The Best Performance".

Addressing two tables, running each classifier with different paraments has stronger impact when using the imbalanced dataset, Table 7, compared with the results from the balanced dataset, Table 6. In details, as Table 6 represents, for the balanced dataset, running J48 and SVM with different parameters result in almost equal outputs for Classification Accuracy, TB, TT, TP rate, FP rate, P,

Figure 3. Time taken to build/test the model using the balanced dataset

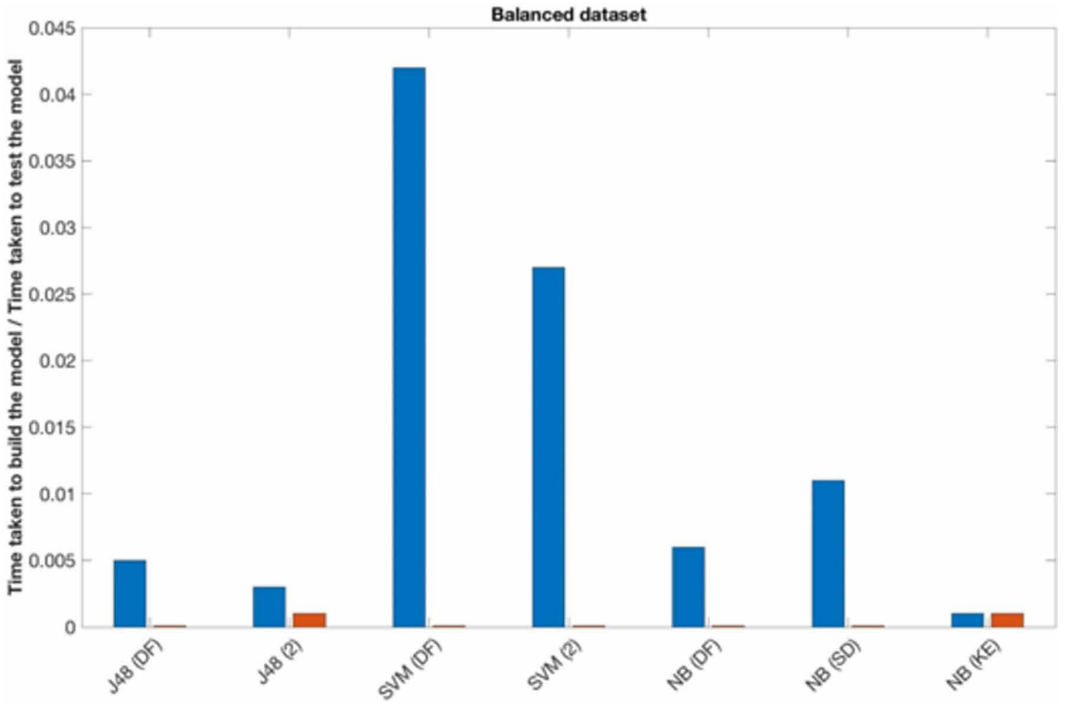


Figure 4. Time taken to build/test the model using the imbalanced dataset

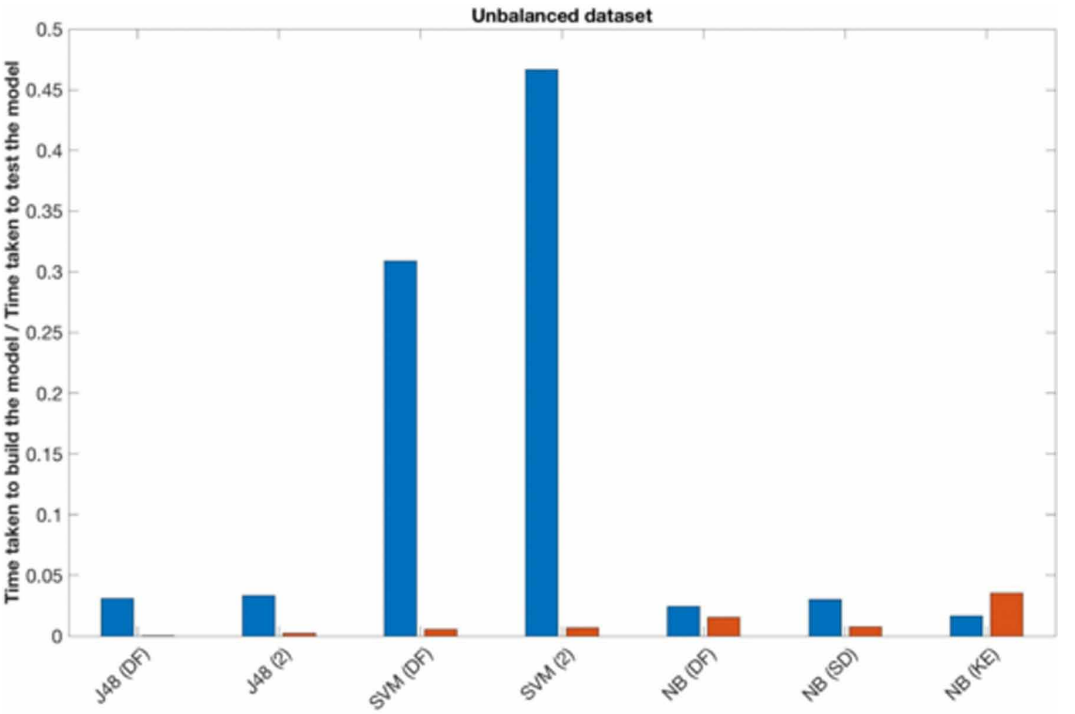


Figure 5. True positive/false positive using the balanced dataset

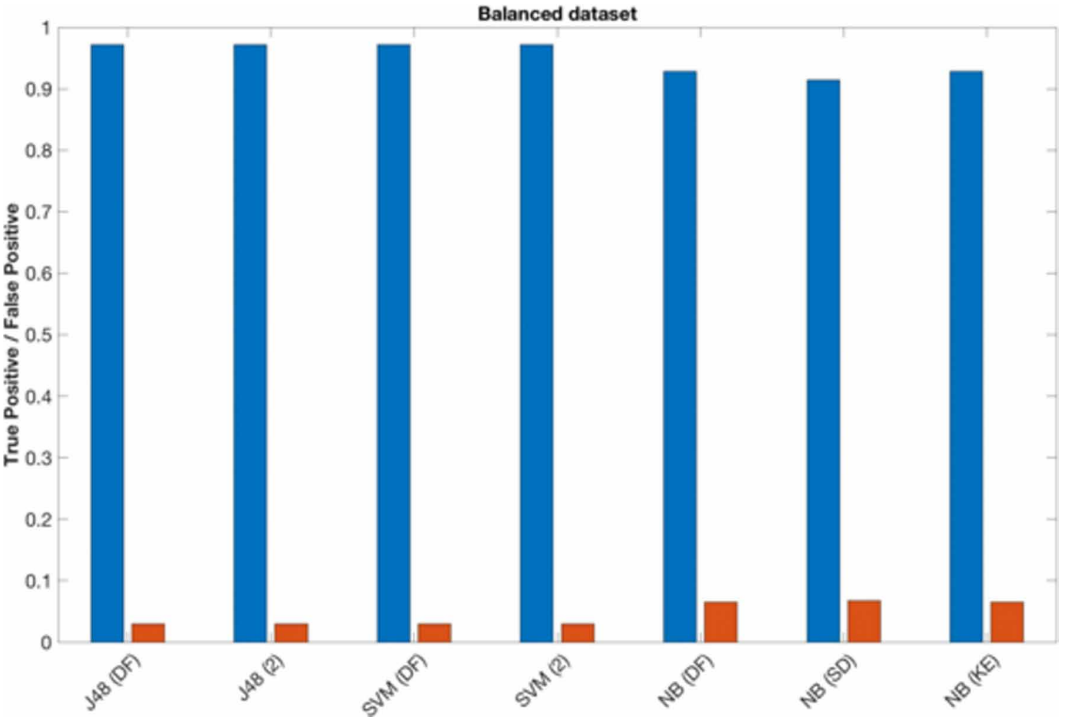


Figure 6. True positive/false positive using the imbalanced dataset

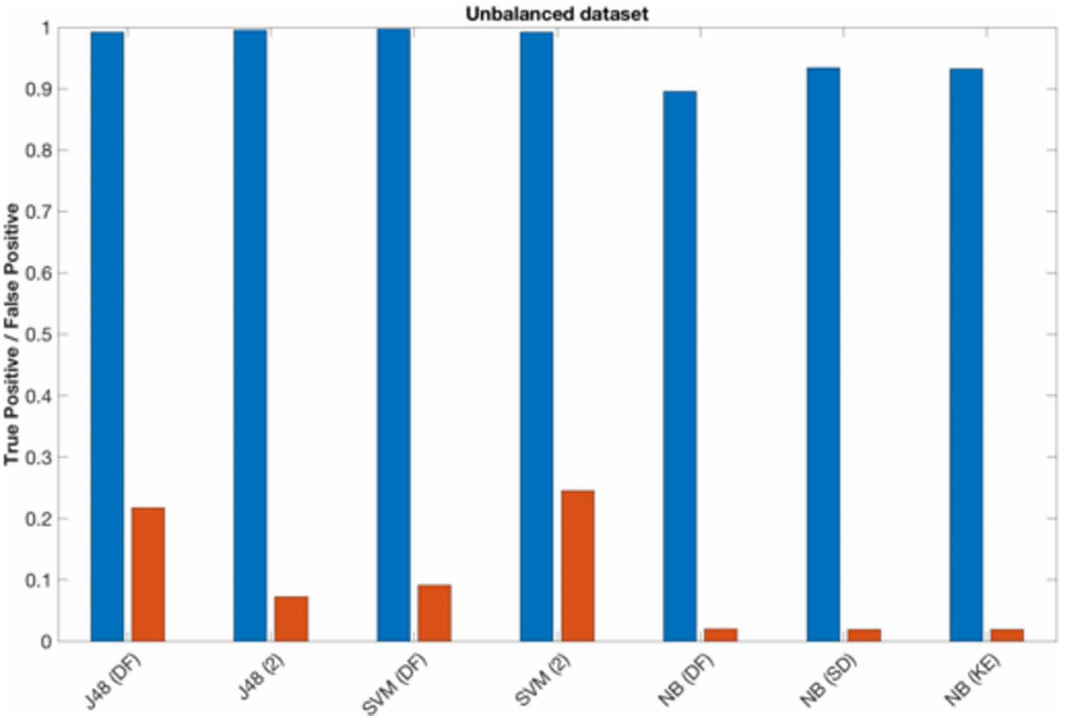


Figure 7. Precision/recall/f-measure using the balanced dataset

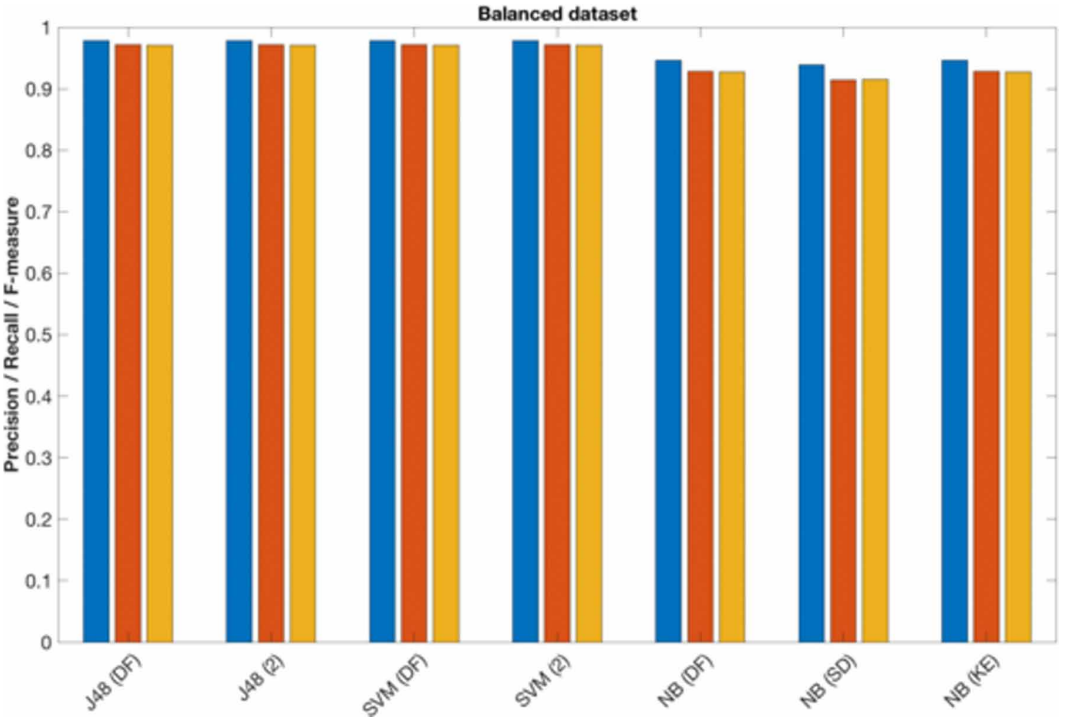
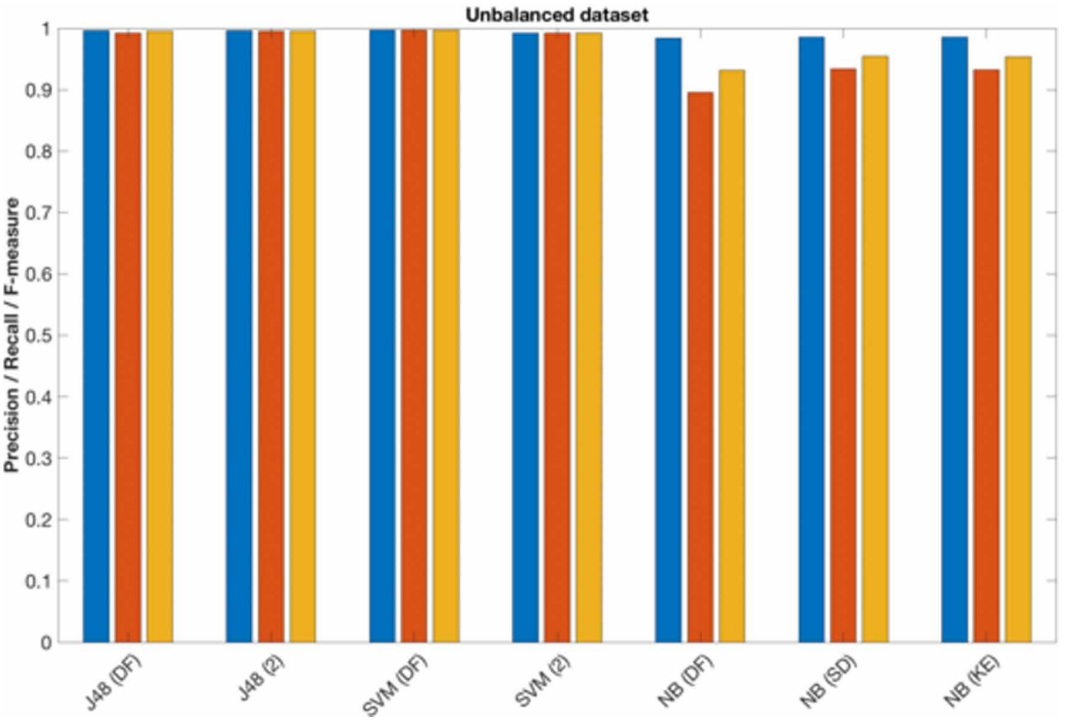


Figure 8. Precision/recall/f-measure using the imbalanced dataset



R, and F each. However, this is not the case for NB where NB (SD) gives the lowest score compared with NB (DF) and NB (KE).

Moreover, as Table 7 represents, for the imbalanced dataset the impact of employing different parameters on CA, TB, TT, TP rate, FP rate, P, R, and F is stronger. For instance, J48 (2) and SVM (DF) perform better than J48 (DF) and SVM (2), both respectively. For NB set of algorithms, NB (SD) performs the best while NB (DF) and NB (KE) are equally worse.

Therefore, answering RQ3, running the classifiers, i.e. J48, SVM and NB, with different paraments will affect the CA, TB, TT, TP rate, FP rate, P, R, and F, however the impact is significantly stronger when using imbalanced dataset compared with the balanced dataset.

5. CONCLUSION AND FUTURE WORK

In this paper, we addressed issues regarding extremely imbalanced datasets with a focus on insider threat problem in which the minority class belongs to the malicious activities and the majority class belongs to the benign activities. For this, we provided a comprehensive review and implementation on the data pre-processing phase which is a vital step in any data mining/machine learning projects. This includes a popular balancing technique known as spread subsample. Additionally, we widely explained six demo setups for the general data breach scenario including: data theft, privileged user data breach, endpoint security processing, shadow IT risk, data security, and sensitive folder breach. We also investigated several parameters for our chosen classifiers and studied the impact of configuring each classifier with these paraments and compared their performance with the default setups. For our experiments, we identified eight performance metrics including: Classification Accuracy (CA), Time taken to Build the model (TB), Time taken to Test the model (TT), True Positive (TP) rate, False Positive (FP) rate, Precision (P), Recall (R), and F-measure (F) along with a few popular machine learning algorithms (i.e. J48 decision tree, SVM, NB, and RF).

We then raised three research questions to answer in this study: 1) the impact of balancing the dataset during the data pre-processing phase (i.e. using spread subsample technique) on the performance metrics mentioned above, 2) the important parameters for the chosen classifiers (i.e. J48 decision tree, SVM, NB, and RF) and 3) the impact of configuring our chosen classifiers with the identified paraments in terms of performance metrics in comparison with the default setups for the balanced and imbalanced datasets. Answering our research questions, we realised that balancing the dataset did not improve CA, TP rate, FP rate, P, R, and F in general but it improved the time taken to build the model and the time taken to test the model. Additionally, we realised that running the classifiers with different parameters affected the performance metrics (i.e. CA, TB, TT, TP, FP, P, R, and F) however the impact was significantly stronger on the imbalanced dataset rather than the balanced dataset.

For our future work, we will investigate all the possible and popular balancing techniques and will implement them on our extremely imbalanced insider threat dataset. We will then analyse and compare their impacts in terms of performance metrics mentioned above (i.e. CA, TB, TT, TP, FP, P, R, and F).

ACKNOWLEDGMENT

The authors would wish to acknowledge the support of the Edinburgh Napier University and The Cyber Academy for funding this work and (ZoneFox, 2017) for providing the demo scenarios.

Table 5. Supervised machine learning results; Classification Accuracy (AC), Time taken to Build/Test the model (TB, TT), True Positive (TP), False Positive (FP), Precision (P), Recall (R), and F-measure (F)

Balanced and Imbalanced datasets	Classification Accuracy		TB TT		TP FP		Precision Recall F-measure	
	B	U	B	U	B	U	B	U
J48 (DF)		*	*			*		*
								*
			*		*			*
J48 (2)		*	*			*		*
								*
			*		*			*
SVM (DF)		*	*			*		*
								*
			*		*			*
SVM (2)		*	*			*		*
								*
			*		*			*
NB (DF)	*		*		*			*
							*	
			*			*		*
NB (SD)		*	*			*		*
								*
			*			*		*
NB (KE)		*	*			*		*
								*
			*			*		*
Total score	**1**	**6**	**14**	**0**	**5**	**9**	**1**	**20**

Table 6. Supervised machine learning results; Accuracy, Time taken to Build/Test the model (TB, TT), True Positive (TP), False Positive (FP), Precision (P), Recall (R), and F-measure (F)

Balanced dataset	Accuracy	TB	TT	TP	FP	Precision	Recall	F-measure	Total score
J48 (DF) J48 (2)	e		*	e	e	e	e	e	1
	e	*		e	e	e	e	e	1
SVM (DF) SVM (2)	e		e	e	e	e	e	e	0
	e	*	e	e	e	e	e	e	1
NB (DF) NB (SD) NB (KE)	e		*		*	*	*	*	5
			*	*					2
	e	*			*	*	*	*	5

Table 7. Supervised machine learning results; Accuracy, Time taken to Build/Test the model (TB, TT), True Positive (TP), False Positive (FP), Precision (P), Recall (R), and F-measure (F)

Imbalanced dataset	Accuracy	TB	TT	TP	FP	Precision	Recall	F-measure	Total score
J48 (DF) J48 (2)		*	*	*					3
	*				*	*	*	*	5
SVM (DF) SVM (2)	*	*	*		*	*	*	*	7
				*					1
NB (DF) NB (SD) NB (KE)				*					1
	*		*		*	e	*	*	5
		*				e			1

REFERENCES

Böse, B., Avasarala, B., Tirthapura, S., Chung, Y. Y., & Steiner, D. (2017). Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams. *IEEE Systems Journal*, *11*(2), 471–482. doi:10.1109/JSYST.2016.2558507

Bradley, N., Alvarez, M., Kuhn, J., & McMillen, D. (2015). IBM 2015 Cyber Security Intelligence Index.

Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys and Tutorials*, *18*(2), 1153–1176. doi:10.1109/COMST.2015.2494502

Cole, E. (2017). *Defending Against the Wrong Enemy: 2017 SANS Insider Threat Survey*. SANS.

Data pre-processing. (n.d.). Retrieved from http://www.cs.ccsu.edu/~markov/ccsu_courses/datamining-3.html

Enterprise, V. (2017). 2017 Data breach investigations report. Zonefox. Retrieved from https://zonefox.com

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews*, *42*(4), 463–484. doi:10.1109/TSMCC.2011.2161285

Gavai, G., Sricharan, K., Gunning, D., Hanley, J., Singhal, M., & Rolleston, R. (2015). Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. *JoWUA*, *6*(4), 47–63.

Gheyas, I. A., & Abdallah, A. E. (2016). Detection and prediction of insider threats to cyber security: A systematic literature review and meta-analysis. *Big Data Analytics*, *1*(1), 6. doi:10.1186/s41044-016-0006-0

Greenberg, S. (1988). Using unix: Collected traces of 168 users.

Hamed, T., Ernst, J. B., & Kremer, S. C. (2018). A survey and taxonomy on data and pre-processing techniques of intrusion detection systems. In *Computer and Network Security Essentials* (pp. 113–134). Cham: Springer. doi:10.1007/978-3-319-58424-9_7

Hanifah, F. S., Wijayanto, H., & Kurnia, A. (2015). Smotebagging algorithm for imbalanced dataset in logistic regression analysis (case: Credit of bank x). *Applied Mathematical Sciences*, *9*(138), 6857–6865. doi:10.12988/ams.2015.58562

Hashem, Y., & Takabi, H., GhasemiGol, M., & Dantu, R. (2016). Inside the Mind of the Insider: Towards Insider Threat Detection Using Psychophysiological Signals. *Journal of Internet Services and Information Security*, *6*(1), 20–36.

Hassabis, D., & Silver, D. (2017). Alphago zero: Learning from scratch. deepMind.

Insiders, C. (2018). Insider threat-2018 report. CA Technologies. Retrieved from https://www.cert.org/insider-threat/tools/

Legg, P. A., Buckley, O., Goldsmith, M., & Creese, S. (2015). Automated insider threat detection system using user and role-based profile assessment. *IEEE Systems Journal*, *11*(2), 503–512. doi:10.1109/JSYST.2015.2438442

Lindauer, B., Glasser, J., Rosen, M., & Wallnau, K. C. (2014). Generating Test Data for Insider Threat Detectors. *JoWUA*, *5*(2), 80-94.

Moradpoor, N., Brown, M., & Russell, G. (2017, October). Insider threat detection using principal component analysis and self-organising map. In *Proceedings of the 10th International Conference on Security of Information and Networks* (pp. 274-279). ACM. doi:10.1145/3136825.3136859

Parveen, P., Evans, J., Thuraisingham, B., Hamlen, K. W., & Khan, L. (2011, October). Insider threat detection using stream mining and graph mining. In *Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing* (pp. 1102-1110). IEEE. doi:10.1109/PASSAT/SocialCom.2011.211

Parveen, P., & Thuraisingham, B. (2012, June). Unsupervised incremental sequence learning for insider threat detection. In *Proceedings of the 2012 IEEE International Conference on Intelligence and Security Informatics* (pp. 141-143). IEEE Press. doi:10.1109/ISI.2012.6284271

Pavlov, D. Y., Gorodilov, A., & Brunk, C. A. (2010, October). BagBoo: a scalable hybrid bagging-the-boosting model. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 1897-1900). ACM.

Ponemon Institute. (2016). *2016 Cost of Insider Threats: Benchmark Study of Organizations in the United States*.

Saarikoski, H. (2011), Explanation of SMO Parameters? Retrieved from https://list.waikato.ac.nz/pipermail/wekalist/2010-December/050570.html

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484.

Singh, A., & Patel, S. (2014). Applying modified K-nearest neighbor to detect insider threat in collaborative information systems. *Ijirset. Com*, *3*(6), 14146–14151.

Torrey, L. & Shavlik, J. (2009). Transfer Learning. In Handbook of Research on Machine Learning. Academic Press.

Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., & Robinson, S. (2017, March). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press.

Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., & Li, K. (2016, April). AI^ 2: training a big data machine to defend. In *Proceedings of the 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)* (pp. 49-54). IEEE Press.

Weka. Class J48. (n.d.). Retrieved from http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

Weka. Class NaiveBayes. (n.d.). Retrieved from http://weka.sourceforge.net/doc.dev/weka/classifiers/bayes/NaiveBayes.html

Weka. Class SMO. (n.d.). Retrieved from http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/SMO.html

Weka. Class SpreadSubsample. Retrieved April 01, 2018, from http://weka.sourceforge.net/doc.dev/weka/filters/supervised/instance/SpreadSubsample.html

Weka. (n.d.). Retrieved from https://www.cs.waikato.ac.nz/ml/weka/

Williams, G., Baxter, R., He, H., Hawkins, S., & Gu, L. (2002, December). A comparative study of RNN for outlier detection in data mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining, 2002* (pp. 709-712). IEEE Press. doi:10.1109/ICDM.2002.1184035

Naghmeh Moradpoor (PhD, MSc, BSc, CEH, FHEA, MBCS) is a lecturer for cybersecurity and networks, the program leader for MSc Advanced Security and Cybercrime, a core member of the Centre for Distributed Computing, Networking and Cybersecurity, and an active member of The Cyber Academy in the School of Computing (SoC) at Edinburgh Napier University. She is a Certified Ethical Hacker (CEH) and an active researcher since 2009. Naghmeh has a research background in machine learning for cybersecurity as well as next generation broadband access networks. This includes Industrial Control System (ICS) and Critical Infrastructure Protection (CIP) security analysis and countermeasures, detection and classification of various cybersecurity attacks for computer networks, data analytics and decision support for cybersecurity, intrusion detection and prevention systems (IDS/IPS) for Internet of Things (IoT), cyberbullying detection and prevention, and forensic similarity hashes. She holds a PhD degree (2013) in Computer Systems and Networks from Ulster University and MSc in Computer Systems and Networks (2009) from Greenwich University, UK. Naghmeh was awarded Outstanding Woman in Cyber at the Scottish Cyber Awards 2017. These Awards are a highly prestigious and flagship event and the first of its kind in Scotland where the best of the best in cybersecurity are being recognised and awarded for their passion, dedication and hard work raising standards in both industry and academia. She has actively participated in leading and organising many cybersecurity events. This includes: Post Graduate Cyber Security (PGCS'16) symposium sponsored by SICSA, Poster Competition for Women in Cyber Security (PCWiCS'17) sponsored by SICSA, Poster Competition in Cyber Security (PCiCS'18) sponsored by SICSA, The First International Conference on Cyber Security and Digital Forensics (CSDF'18) sponsored by SICSA NEXUS, and The First/Second International Workshop on Securing IoT Networks (SITN'17 & '18). Naghmeh has been an invited speaker for Women in Science Festival - Dundee (2015), Critical Infrastructure Protection in SEE-IT Summit (Cybersecurity track) - Serbia (2018) sponsored by SICSA and SICSA NEXUS, and Gender Equality & Women in Cybersecurity in SEE-IT Summit - Serbia (2018) sponsored by SICSA and SICSA NEXUS. She has been invited to serve as a program Chair, technical program committee member and a session Chair of major international conferences and an editorial board as well as a reviewer for prestigious journals since 2013. Some of these conferences and journals include: 7th to 11th international conference on Security of Information and Networks (SIN), Special Session on Industrial Automation and Process System's Security (ISIE-IAPSS'17) in conjunction with IEEE ISIE'17, The First/ Second International Workshop on Secure Smart Societies in Next Generation Networks (SECSOC'17 & '18) in conjunction with IEEE iThings'17 & '18. Journal of Information Security and Applications (Elsevier), Journal of Computers & Security (Elsevier), Journal of Cyber Security Technology, IEEE Access Journal, and Journal of Optical Switching and Networking (Elsevier). In addition, Naghmeh has been involved in supervising Honours, summer interns, MSc, and PhD students since she achieved her PhD in 2013. Naghmeh is a Director of Studies (DoS) for two collaborative PhDs. The first one is a PhD research work on ICS & CIP security analysis and countermeasures using machine learning with a focus on clean water supply systems. It is an ongoing and successful collaborative research work between Edinburgh Napier University (School of Computing) and Edinburgh Napier University (School of Engineering & Built Environment). Secondly, Naghmeh acts as a DoS for a PhD project on cyberbullying detection using victim's sentiment analysis in social media. It is also a current collaborative and successful research work between Edinburgh Napier University (School of Computing) and King Abdulaziz University in the Kingdom of Saudi Arabia. Naghmeh and her students (Honours, MSc and PhD) have produced significant and ongoing joint research publications.