applied sciences

Article

Insider Threat Detection Using Machine Learning Approach

Bushra Bin Sarhan *, Najwa Altwaijry

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia
* Correspondence: 439204602@student.ksu.edu.sa

Abstract: Insider threats pose a critical challenge for securing computer networks and systems. They are malicious activities by authorised users that can cause extensive damage, such as intellectual property theft, sabotage, sensitive data exposure, and web application attacks. Organisations are tasked with the duty of keeping their layers of network safe and preventing intrusions at any level. Recent advances in modern machine learning algorithms, such as deep learning and ensemble models, facilitate solving many challenging problems by learning latent patterns and modelling data. We used the Deep Feature Synthesis algorithm to derive behavioural features based on historical data. We generated 69,738 features for each user, then used PCA as a dimensionality reduction method and utilised advanced machine learning algorithms, both anomaly detection and classification models, to detect insider threats, achieving an accuracy of 91% for the anomaly detection model. The experimentation utilised a publicly available insider threat dataset called the CERT insider threats dataset. We tested the effect of the SMOTE balancing technique to reduce the effect of the imbalanced dataset, and the results show that it increases recall and accuracy at the expense of precision. The feature extraction process and the SVM model yield outstanding results among all other ML models, achieving an accuracy of 100% for the classification model.

Keywords: insider threat; deep learning; anomaly detection

check for updates

Citation: Bin Sarhan, B.; Altwaijry, N. Insider Threat Detection Using Machine Learning Approach. *Appl. Sci.* **2023**, *13*, 259. https://doi.org/10.3390/app13010259

Academic Editor: Giacomo Fiumara

Received: 27 November 2022
Revised: 14 December 2022
Accepted: 20 December 2022
Published: 25 December 2022

1. Introduction

Cybersecurity attacks are becoming more frequent, targeting widespread domains with frightening consequences. Attacks today are carried out using trending technologies that are hard to detect. The main goal of cybersecurity is to protect organisations and individuals from cyber attacks and prevent or mitigate harm to computer networks, applications, resources, and data. Cyber defence mechanisms exist at the application, network, host, and data levels. Industry surveys show that 79% of security threats are insider threats, i.e., malicious acts carried out by the organisation's careless or disgruntled employees who abuse their authorised access to networks, systems, and data [1]. Insider threats can cause extensive damage, so defenders must guard against them. Identifying authorised users who are harming the organisation while they are trusted is the most difficult cybersecurity challenge [2].

Insider threat is a substantial problem in the cybersecurity field. As insiders act on the system as regular users, it makes the threat more difficult to detect and classify. Previous studies on company's internal security have mainly focused on detecting and preventing outside intrusion. Although several studies have tried to address the issue of detecting insider threats using machine-learning approaches, there is still room for improvement. The number of studies developed using current advanced machine learning algorithms is low. An insider threat is defined as "a current or former employee, contractor, or other business partner who has or had authorised access to an organisation's network, system, or data and who intentionally (or unintentionally) exceeds or misuses that access to negatively affect the confidentiality, integrity, or availability of the organisation's information or information systems" [3]. Insider threat has become a widespread issue and a significant challenge in cybersecurity. There are four main types of cyber insider threats:

1. Data theft: where the attacker attempts to steal private information. Examples of activities that indicate this scenario include: accessing sensitive data at odd hours, downloading data to personal devices, or sending data outside the protected perimeter.
2. Privilege abuse: a particularly difficult to detect insider attack that involves users with privileged access rights uploading harmful data or editing/deleting activity logs, creating privileged accounts without a request, deploying suspicious software, or changing security configurations.
3. Privilege escalation: where a regular user attempts to gain privileges to abuse resources. Indicators of such as attack include frequent and unnecessary access requests, showing unusual interest in data and projects that a user cannot access, or installing unauthorised software.
4. Sabotage: where the employee attempts to destroy data or infrastructure. Some indicators of this attack include sending emails with attachments to competitors, deleting accounts intentionally, failing to create backups, or making changes to data that no one requested.

In this study, we focus on the first type: Data theft [4]. Data theft can be detected by monitoring user activity using a set of rules put in place by the organisation's security officers. However, if a rule does not exist, then the user will not be reported. Another method is active threat hunting by the security officers. Both of these methods require human expertise to detect attacks. A newer approach is to use a user and entity behaviour analytics tool, which detects a deviation from a user's normal behaviour using a machine learning approach.

In order to detect malicious users, detailed usage logs within the organisation are collected and analyzed. The number of potential features we can develop from the logs grows rapidly when we access detailed information on the insiders. Developing concepts, manually building software, and extracting features may be time-consuming and may need adaptation each time a log is changed or updated. This research utilises an automated approach, known as deep feature synthesis (DFS) to quickly build a comprehensive collection of features that characterise a specific employee's use behaviours. The number of features detected will usually be very large and may lead to the overfitting problem in machine learning when the number of features far exceeds the number of data examples. As such, we use a feature dimensionality reduction algorithm to select the most important features, and then use the data to train our detection system.

Detecting anomalies aims to uncover any strange behaviour on the computer system and catch risky user behaviour before data are lost. It uses acceptable or usual behaviours in the system and quantifies them in an attempt to isolate irregular behaviour, categorising them as intrusive or not. The system will detect anomalous behaviour in an organisation by using machine learning algorithms that detect anomalies (abnormal activities) in user behaviour to ensure that the security of the system is maintained and information is kept safe. Thus, the aims of this study are threefold: (1) to develop an intelligent machine learning model utilising the most recent advances in machine learning approaches coupled with feature engineering to detect anomalies of potential insider threats based on the CERT insider threat dataset, (2) to ensure that the system pipeline is streamlined and designed to fit into any organisational log file structure, and (3) that the system has the capability to easily adapt to changes to log file structure. Such a system would be able to detect zero-day attacks in real time. The system has the ability to recognise anomalous user behaviour compared with the user baseline. Once the system is installed on the network, it can be used to monitor incoming user actions that are written on the corresponding logs and send appropriate messages to the security officer. A new user added to the system will require the training phase to be repeated in order to enrol the user into the system. An important consideration in cyber security nowadays is to build a system with strong guarantees of resiliency in response to attacks. Formal verification is an approach used to prove a system's compliance with the required security properties [5–7]. Machine learning algorithms may be verified during both the data preparation and training phase, as well as the verification of the machine learning system itself [8,9]. Currently, research efforts are still far away from being able to verify the entire machine-learning pipeline [10]. However, the formal verification of random forests, SVMs, and NNs have been studied [10–12]. Before the system is deployed in an organisation, formal verification of the system should be conducted, both for the data pre-processing and training stage, as well as verification of the system itself.

The rest of this paper is organised as follows. Section 2 reviews and evaluates existing anomaly detection and classification machine learning approaches. Section 3 describes the utilised CERT dataset and gives our proposed system's overall methodology. Section 4 shows and discusses the results of ML models used to solve the insider threat problem. Finally, we conclude with the conclusion in Section 5.

2. Literature Review

Insider threat detection is a broadly researched topic; a variety of solutions have been proposed: specifically, different learning techniques to facilitate early, more accurate threat detection. To discover present research gaps and potential future research domains, an analytical review of the various approaches to insider threat identification is required.

Over the past two decades, researchers have investigated insider threat detection and prevention using anomaly-based approaches. These techniques "learn" from normal data only to detect anomalous instances that deviate from expected instances; this approach has remained the most popular method in the literature. Anomaly-based detection is based on one major assumption: that an attacker's actions differ from normal patterns of actions. Specifically, some of the common behaviours associated with insider threats include (i) the collection of large datasets and (ii) uploading files that originate from outside the organisation's website [13]. One crucial shortcoming of this traditional approach to anomaly detection is that once the baseline has been fully modelled, anything outside this threshold will be considered a potential threat; this causes an abundance of false positives [14]. Moreover, classification-based insider threat detection represents an alternative research method; it "learns" from normal and anomalous data to determine the decision boundary that distinguishes normal from anomalous incidences.

This section provides an up-to-date, comprehensive survey of recent approaches that address insider threat detection: (i) machine learning (ML) and deep learning (DL) approaches (either anomaly-based [13–27], and (ii) classification-based approaches [2,14,25,28–42]).

2.1. Machine Learning

Researchers have employed many different algorithms for the insider threat detection problem, such as deep neural networks [43], multi-fuzzy classifiers [37], hidden Markov method [41,44], one-class support vector machines [40], deep belief networks [43], linear regression [26], clustering algorithms [24], and light gradient boosting machine [36]. We outline some of the more significant studies below.

A study by Noever [2] investigated different families of ML algorithms; the findings suggest that random forest offers the best results compared with other ML models. The experiments were conducted using the CERT insider threat dataset, and a feature vector was produced by extracting the risk factors from the data. They incorporated sentiment analysis factors from email and website content and file-access details. In general, they ranked these features based on their importance.

Another interesting ML method that has gained attention is iForest [16,17]. Gavai et al. [17] used iForest as an unsupervised anomaly detection method; they identified statistically abnormal behaviour using features extracted from social data, including email communication patterns and online activities. With this strategy, they take advantage of the fact that insider threats are more likely to be started by workers who intend to leave the organisation. The authors used iForest to predict when employees would quit the company as a proxy to identify the likelihood of insider threats; they obtained a ROC score of 0.77 for insider threat detection. Karev et al. [16] also used iForest for insider threat detection but on an online framework. A generic algorithm was used to find the optimal HTTP features to help detect abnormal insider behaviours from normal behaviours. Although the study achieved good results (82% AUC), it was limited by its use of HTTP log data only in the CERT dataset.

Employing a predictive model that uses a ML algorithm on an imbalanced dataset has been shown to produce high levels of inaccuracy and bias. The lack of real-world data and the issue of data imbalance mean that insider threat analysis remains an understudied research area. Various researchers have addressed the pre-processing step of balancing the dataset [14,29,34,39]. The results obtained by Sheykhkanloo and Hall [30] showed that no significant improvement in performance was achieved when using the spread-subsample technique to balance datasets. However, the method significantly reduced the amount of time needed to build and test the model. Additionally, their experiments showed that all supervised ML algorithms, except Naïve Bayes, achieve better performance for imbalanced datasets.

Orizio et al. [22] utilised a constraint learning algorithm for insider threat detection. The algorithm creates an optimised constraint network that represents normal behaviour; it detects threats when the cost exceeds a specified threshold. One advantage of this approach is that it provides an explanation for making the decision—unlike most other ML algorithms. They manually select features and suggest using deep learning models in the feature extraction process to enhance the results.

Gayathri et al. [35] considered the problem of insider threat detection using a deep learning approach; their approach combines a generative model with supervised learning to perform multi-class classification. They used Generative Adversarial Network (GAN) on the CERT insider threat dataset for data resampling to enrich the minority data samples. To choose GAN, three different resampling techniques were used on four different classification methods; the GAN method was nominated due to its promising results compared to the other resampling techniques.

2.2. Deep Learning

Traditional shallow ML models are unable to fully utilise user behaviour data despite existing approaches demonstrating excellent performance on insider threat detection because of their high dimensionality, complexity, heterogeneity, and sparsity [45]. Deep learning, on the other hand, can be an effective tool for examining user behaviour in an organisation in order to spot malicious insiders. Deep learning is a representation learning algorithm that can learn multiple levels of hidden representations from complex data based on its deep structure [45].

Several methods for detecting insider threats have recently been proposed, deep feed-forward neural networks, convolutional neural networks, and recurrent neural networks. This section introduces some contemporary deep-learning methods for the detection of insider threats.

2.2.1. Recurrent Neural Networks

The use of traditional techniques, such as cumulative sum and exponentially weighted moving averages, for monitoring data logs has proven to be challenging. In conventional log data monitoring systems, long data sequences are common; this results in longer processing times; therefore, there is a need to achieve pre-determined monitoring results and detection, which the aforementioned approaches are unable to achieve. These approaches need to feature a pre-determined time window in the detection of significant changes in an underlying data distribution. One significant benefit of RNNs is that stacking deep layers in RNNs allows the network to learn over different time scales [46].

Many researchers have employed LSTM as an unsupervised anomaly detection method, such as [15,21,27]. This approach is meant to overcome the challenges of RNNs, such as the reduction in time gradients in long data sequencing. The inefficiencies associated with the use of RNNs were also highlighted by [47]; namely, this technique offers poor stability, especially when applied to a long sequence of imbalanced data. Large training datasets and an imbalanced distribution in such training sets in most practical cases of anomaly detection make RNN a less favourable approach.

Meng et al. [19] suggested using LSTM-RNNs and Kernel PCA to analyze insider threats. CMU CERT Insider Threat dataset v6.2 was used to build and test the model. Performance comparison evaluation of the proposed technique was conducted against traditional ML algorithms (e.g., SVM and isolation Forest); it was not compared against deep learning models. This approach achieved a TPR of 92.46%, an FPR of 6.8%, a precision of 95.12%, and an accuracy of 93.85%. This accuracy was achieved through thoroughly pre-processing the initial log data. First, events were standardised and aggregated into a format around the behaviours and attributes of individuals; next, features were extracted for the training and testing phases, respectively.

Tian et al. [23] proposed an insider threat detection method based on an attention-LSTM that models normal user behaviour and indicates anomalies as malicious behaviour. They used a multi-head attention-LSTM (that has shown high priority in neural language processing) to separate the attacked data from normal data in parallel, given different types of features. The Dempster–Shafer theory was then employed to determine whether a given set of input data qualifies as an attack.

A different approach proposed by Rastogi et al. [38], known as DANTE, uses system logs to generate a sequence of events for each user over a given time frame to establish ground truth. Their model uses LSTM to process the sequences and classify normal and outlier behaviours. They further classify threats into one of five categories provided in the CERT insider threat dataset. Although the model achieves high accuracy, it suffers from a high FP rate.

2.2.2. Convolutional Neural Networks

Many researchers have applied CNN in their proposed work, such as [14,27,33,35]. An interesting approach to solving the problem of insider threat detection was introduced by Gayathri [39]. Here, image classification was used to classify insiders. They used the undersampling technique prior to extracting the features. After the extraction of 20 features, they generated a grey-scale image for each user within a day. The grey-scale image was then fed into a deep pre-trained CNN for classification. The main focus of the paper was the transference of learning on MobileNetV2, VGG19, and ResNet50 pre-trained models; the results were promising. Another work by Koutsouvelis [29] used a CNN trained from scratch to classify coloured images that represent either malicious or normal activities. Their results were promising, although they used a long-time window where each image represents weekly or monthly activities.

Yuan et al. [45] further showed the attainability of a hybrid, high-performance anomaly detection model by combining CNN and LSTM for deep neural networks (DNN). Their work is focused on a feature engineering process to detect insider threats. First, LSTM is used to extract and abstract the temporal features of users and then learn user behaviour and language through their actions on a network. The second stage is to convert the features extracted by LSTM to fixed-size matrices and then apply CNN algorithms to the matrices to detect anomalies and insider threats. The results show the successful detection of insider threats and anomaly detection but fail to present data on accuracy and FP rates. Moreover, the method suffers from delays in the conversion of features to fixed-size matrices, which affects its overall performance.

Singh et al.'s [31] approach detect insider threats using a different approach: user behaviour profiling; this observes and explores user behaviour action sequences. The researchers presented a hybrid ML model comprised of convolution neural networks (CNN) and multi-state long short-term memory (MSLSTM) to identify a stabiliser outlier in the behavioural patterns. Deep neural networks were applied in two stages: first, the

LSTM samples user behaviour temporal action sequences to extract the temporal features and encode each user behaviour action sequence. LSTMs are able to process any size input and output length; these features are then converted into a fixed-size low-dimensional real vector matrix. The fixed-size feature matrices are given as input to the CNN to detect insider threats; the matrices take only fixed-size inputs and outputs to determine the particular action sequence of a particular user as either normal or anomalous. Multi-state LSTM provided accuracy rates of 0.9042 and 0.9047 on training and testing data, respectively. Given the experimental results, multi-state LSTM is preferred for single-state LSTMs.

The previous literature review explained several techniques used by different researchers for anomaly detection and highlighted the complexity of tackling insider threats: mainly because, unlike other types of cyberattacks, the internal attacker has access and privileges to information systems; hence, they do not have to bypass existing IDS or firewalls. From the findings, it is evident that insider threats are eminent and can result in catastrophic ramifications for a multitude of diverse organisations. Feature extraction of user behaviour is an essential consideration and foundation of anomaly detection. However, using deep learning for insider threat detection still faces various challenges related to the characteristics of insider threat detection data, such as extremely small incidences of malicious activities and adaptive attacks. Hence, developing advanced deep learning models that can improve the performance of current state-of-the-art insider threat detection solutions remains an under-explored research area. There is a general lack of research on feature-engineering-based deep learning techniques for insider threat detection. Future research could address the possibility of using a hybrid ML approach and leveraging data mining for feature extraction as a means of reducing FP rates. The literature review has highlighted that most of the existing methods suffer from inefficiencies, especially when using a large dataset consisting of imbalanced data, which provides an opportunity for further research. In this study, we propose a solution to fill this gap using the CERT Insider Threat dataset.

Table 1 summarises the most significant results from the literature review. All apply their methods with different versions of the CERT insider threat dataset using either supervised or unsupervised machine learning algorithms. The anomaly detection method produced the best results, with an accuracy of 99.91% with a very low false positive rate, while the best results achieved by classification methods were an accuracy of 99% and a precision and recall of 99.32%. Anomaly detection studies had no need for balancing techniques, and three classification studies employed balancing techniques with two forgoing balancings.

Table 1. Literature review summary. Abbreviations: Accuracy (acc), False Positive Rate (FPR), False Alarm Rate (FAR), Precision (P), F1 − Score (F1), Recall (R). The learning method is either supervised (S) or unsupervised (U). The data column shows the version of CERT used in the study.

Ref	Method	Learn	Data	Results	Type	Balancing
[2]	Random forest	S	r4.2	acc: 98%	Classification	Under/Over sampling
[16]	iForest	U	-	AUC: 82%	Anomaly detection	-
[22]	constraint network	U	r4.2	acc: 99.91% FPR 0.06% P: 99.84% F1: 55.00%	Anomaly detection	-
[39]	CNN	S	r4.2	acc: 99% P: 99.32% R: 99.32%	Classification	Random under sampling
[19]	LSTM	U	r6.2	TPR: 92.46% FPR: 6.8% P: 95.12% acc: 93.85%	Anomaly detection	-
[23]	attention-LSTM	U	r6.2	R: 95.79% FAR: 4.67% acc: 95.47%	Anomaly detection	-

Table 1. *Cont.*

Ref	Method	Learn	Data	Results	Type	Balancing
[38]	RNN-LSTM	U	r6.2	acc: 93%	Classification	-
[35]	XGBoost RF, MLP, 1DCNN	S	r4.2	P: 83% R: 76%	Classification	GAN
[23]	attention-LSTM	U	r6.2	R: 95.79% FAR: 4.67% acc: 95.47%	Anomaly detection	-
[37]	CNN-LSTM	S	r6.2	AUC: 90.47%	Classification	-

3. Materials and Methods

This study aims to build an advanced insider threat detection system by leveraging modern machine learning algorithms. The preparation of the dataset is essential in the implementation of any machine learning algorithm. Hence, it is crucial to prepare the data by cleaning and pre-processing the raw data to be fitted to the learning algorithm. Once the dataset is cleaned and pre-processed, feature extraction extracts the latent patterns that could help the designed algorithm learn as independent variables. Then, advanced machine learning algorithms are utilised to develop a comprehensive model that can classify a given data point with high accuracy as being either malicious or a regular activity. To measure the model's accuracy, we will use evaluation metrics accuracy, recall, precision, and F1 − Score. Figure 1 illustrates the research methodology.

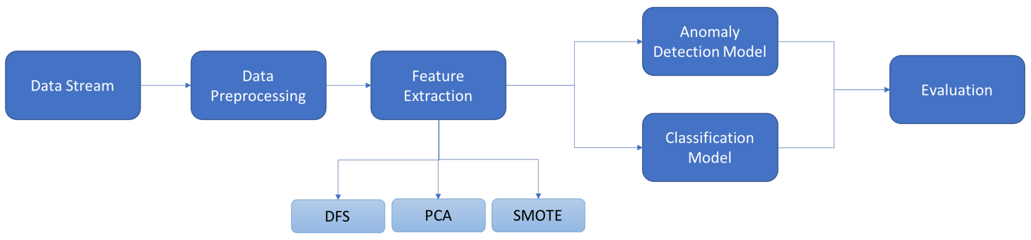


Figure 1. Research methodology.

3.1. Dataset

The lack of actual data is a major barrier for researchers studying the insider threat problem. These data involve log files that contain private user information [3]. In order to protect their users and assets, organisations frequently deny researchers access to real data. However, under specific regulations, an organisation may agree to give the researchers restricted access after anonymizing the private and confidential attributes of the data.

This problem makes it difficult for the researchers to continue their work. In order to solve the insider threat detection problem, it is, therefore, preferable to use synthetic data in the system's design and evaluation. The DARPA ADAMS [48] and Schonlau datasets [49], among others, have been used in research papers in the past, even though they were not specifically created to address the insider threat issue. However, these datasets are less useful due to the complexity of the insider threat problem.

The CMU-CERT dataset(s) has seen a significant increase in usage for insider threat detection systems over the past decade. The CERT insider threat dataset [50] is a collection of artificial datasets produced by the Community Emergency Response Team (CERT) at Carnegie Mellon University (CMU) [51]. It is widely used in insider threat detection studies [43]. There are several releases of the CERT insider threat dataset, and the most used versions are r4.2 and r6.2. Table 2 shows the two versions' statistics, demonstrating that compared to prior CERT dataset versions, CERT r4.2 has a higher rate of malicious activities.

Table 2. CERT datasets r4.2 and r6.2 statistics.

Version	Employees	Insiders	Activities
r4.2	1000	70	32,770,227
r6.2	2500	5	135,117,169

Thus, we used the CERT r4.2 dataset, which consists of five different events—logon/logoff events, email transmission, devices, files, and HTTP events—as illustrated in Table 3. These events document the activities of 1000 employees in an organisation over a period of 17 months. The dataset has 32,770,222 events from 1000 normal and abnormal users. Experts intentionally injected 7323 malicious insider instances. The CERT dataset contains the psychometric score for each employee, also known as the "Big Five personality characteristics". These characteristics are stored in the psychometric.csv file.

Table 3. Dataset files and features.

File	Feature Description
logon.csv (logon/logoff activities)	ID, date, user, PC, activity
device.csv (external storage device usage)	ID, date, user, PC, activity (connect/disconnect)
email.csv (email traffic)	ID, date, user, PC, to, cc, bcc, form, size, attachment count, content
http.csv (HTTP traffic)	ID, date, user, PC, URL, content
file.csv (file operations)	ID, date, user, PC, filename, content
psychometric.csv (psychometric score)	ID, user, openness, conscientiousness, extraversion, agreeableness, neuroticism

In terms of insider threats, Version r4.2 of the dataset consists of three primary scenarios described as follows:

1. A user who never worked after hours or used removable drives starts logging in after hours, using a removable drive, uploading information to wikileaks.org, and then leaves the company shortly after;
2. A user begins searching for career opportunities on job search websites and contacting potential employers. Before leaving the office, they use a thumb drive to take data (at a rate noticeably higher than their prior actions);
3. A dissatisfied system administrator downloads a key logger and transfers it to his supervisor's computer using a thumb drive. The following day, he logs into his company's network as his boss and sends out an alarming mass email, causing widespread concern, and immediately leaves the organisation.

As already indicated, we have focused mainly on the CERT r4.2 dataset because the CERT r4.2 dataset contains a high number of insider threats compared with previous and later versions. Our attention is drawn to the first scenario, which corresponds to a data theft attack, as compared with privilege abuse, privilege escalation, or sabotage attack. We extrapolate data from the files device.csv, http.csv, psychometric.csv, and logon.csv. The description of the scenario includes the following: hours worked (logon.csv), using a removable device (device.csv), and uploading information (http.csv). We exclude file.csv and email.csv due to runtime considerations when using DFS, as outlined in Section 3.3.1.

3.2. Data Pre-Processing

Data collection and pre-processing are essential not only to detect insider threats but also to perform other cybersecurity activities. Successful implementation of machine learning techniques is possible by following suitable processing steps combined with enough data collection. This allows security analysts to make correct conclusions with the help of machine learning techniques. The data collected often lack the background information necessary for feature extraction. Hence, we performed a feature engineering step in data pre-processing. Utilizing feature engineering, we were able to gather valuable supplementary data for additional processing to determine normal business hours and website categories. Table 4 describes the added features.

Table 4. Extra features added to enhance the feature extraction process.

Feature Name	Feature Description
Weekday_Logon_After	Employees log on outside of working hours on weekdays
Weekday_Logon_Normal	Employees log on during working hours on weekdays
Weekend_Logon	Employees log on during weekends
url_Count	The number of URLs visited by an employee in a day

3.3. Feature Extraction

One of the main problems with insider threat detection is the extraction of features throughout the feature engineering process. There is no rule regarding the number of features derived from each log source, and it has been found to be variable in different studies. The CERT insider threat dataset is seen as a relational dataset from which features may be manually derived from the relationships between the entities (files). All researchers try to manually derive as many features as possible to glean high-quality descriptive results. One powerful automated feature engineering tool that has gained the attention of many scholars working in relational datasets over the last couple of years is Deep Feature Synthesis (DFS). This algorithm captures features usually supported by human intuition and performs feature engineering for multi-table and transactional datasets commonly found in databases or logs. After the DFS process, the data are gathered so that every event relating to a user is represented in a separate feature vector. We build 1000 user-specific profiles based on user action sequences.

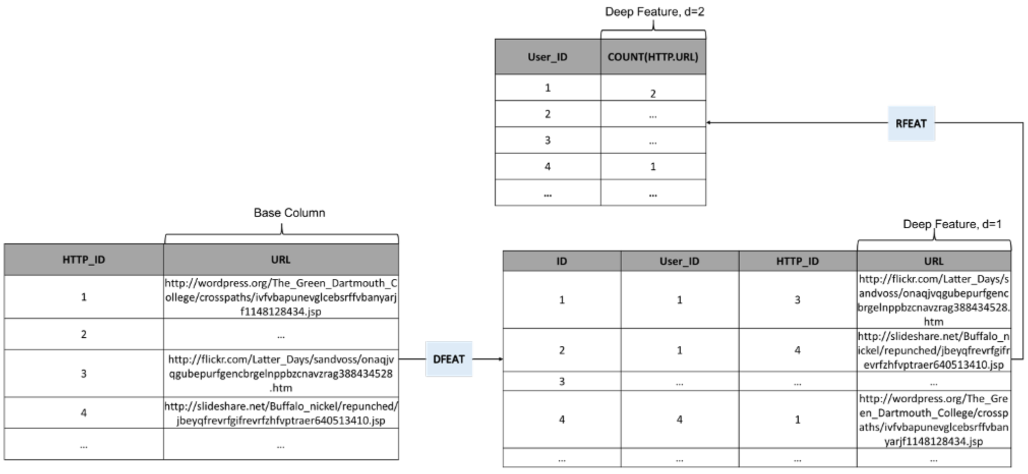
Given that many of the DFS characteristics are categorical and hence unreadable by the ML methods we are using, the aggregated data must be transformed into the right format. We perform one-hot encoding for categorical values. Further, all columns containing null values are removed. DFS results in 69,737 features. DFS produces a large number of features that lead to the dimensionality curse issue (where the dimensions are complex and difficult to visualise). To determine the most important characteristics in the CERT dataset that have the greatest influence on the target variable, we employed PCA as a dimensionality-reduction technique. It produces a reduced set of the most important linear combinations of the initial set of features, which are uncorrelated features. After PCA, we obtain a final 553 features. We used SMOTE [52] balancing techniques because CERT is an imbalanced dataset, as the anomalous samples are significantly less than the standard samples. DFS, PCA, and SMOTE are described in the following subsections.

3.3.1. Deep Feature Synthesis

Machine learning algorithms rely heavily on their input features. A suitable choice of features is an essential process in any machine learning algorithm; however, it must involve human intuition in many cases. At the same time, recent developments in deep learning algorithms can omit the choice of a suitable feature because the feature is learned through the network architecture. However, feature selection remains a human-intuition-driven and time-consuming phase in other machine learning algorithms.

Previous researchers have not considered automated feature engineering in their work before. This study used an automated feature engineering tool to improve the effectiveness of insider threat detection approaches. The DFS tool takes into account many of the manually selected features from prior research, as well as many more additional features. DFS was proposed by J. M. Kanter and K. Veeramachaneni [53]. It automatically creates features for relational datasets based on connections in the data to a base field, generating predictive models from raw data. Mathematical functions were then sequentially applied along that path to constructing the final features, as illustrated in Figure 2. It implements a generalisable machine-learning pipeline tuned using a novel Gaussian Copula process-based approach [53].

The algorithm forms some specialised features from the given data by intuiting what could predict the outcome. The algorithm catches features that are generally aided by human interpretation, although it is automatic. It performs feature engineering for various tables and transactional data found in databases and log files [53]. This approach helps data scientists save time by deriving new features that are specific to the dataset and applying the same mathematical equations. The features created by DFS are easier to understand since they are based on primitive combinations that can be easily described in natural language; this helps the data scientist understand the features generated by DFS.



User_ID	COUNT(HTTP.URL)
1	2
2	...
3	...
4	1
...	...

HTTP_ID	URL
1	http://wordpress.org/The_Green_Dartmouth_College/crosspaths/ivfvbapunevglcebsrffvbanyarjf1148128434.jsp
2	...
3	http://flickr.com/Latter_Days/sandvoss/onaqjvqgubepurfgencbrgelnppbzcnavzrag388434528.htm
4	http://slideshare.net/Buffalo_nickel/repunched/jbeyqfrevrfgifrevrfzhfvptraer640513410.jsp
...	...

ID	User_ID	HTTP_ID	URL
1	1	3	http://flickr.com/Latter_Days/sandvoss/onaqjvqgubepurfgencbrgelnppbzcnavzrag388434528.htm
2	1	4	http://slideshare.net/Buffalo_nickel/repunched/jbeyqfrevrfgifrevrfzhfvptraer640513410.jsp
3	...	...	...
4	4	1	http://wordpress.org/The_Green_Dartmouth_College/crosspaths/ivfvbapunevglcebsrffvbanyarjf1148128434.jsp
...	...	...	...

Figure 2. An illustration of a feature that DFS can produce.

The inputs of this algorithm are entity sets with several data types, such as numeric, categorical, and timestamps, a set of relationships, and several mathematical functions. The mathematical functions are applied at two different levels: the entity level and the relational level. DFS generates three sorts of features: Entity features (efeat), Direct Features (dfeat), and Relational features (rfeat), as illustrated in Figure 3 [53].

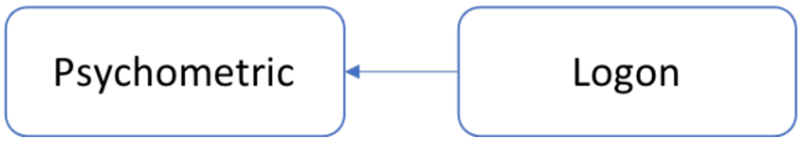


Figure 3. Demonstration of a relationship.

Entity features (efeat) are calculated by considering only one entity, such as translating an existing feature into another value type, converting a categorical string data type to a pre-decided unique numeric value, or rounding a numerical value. An example of such a computation is a cumulative distribution function (cdf)-based feature. To generate this feature, the DFS forms a density function over $x_{:,j}$, and then evaluates the cumulative density value for $x_{(i,j)}$ (or percentile), thus forming a new feature [53].

When there is a forward relationship between one instance of entity E_I, instance m, and one instance of entity E_k, instance i, i has an explicit dependency on instance m. In Figure 4, an e-commerce example, the Logon entity has a forward relationship with the Psychometric; each logon is related to only one user in the Psychometric table. Direct features (dfeat) are applied over the forward relationships. In these, features in related entity $i \in E_k$ are directly transferred as features for the $m \in E_I$ [53].

For backward relationships, from instance i in E_k to all instances $m = 1 \dots M$ in E_I that have a forward relationship with k. In the same example above, the Logon entity has a backward relationship with Psychometric; many logons can point to the same user in the Psychometric table. The functions min, max, and count are examples of rfeat functions. Other rfeat functions include functions that could be applied to the probability density function [53]. The depth of a DFS is based on how early the decision tree or other framework response is activated [53]. Without DFS, scientists require coding to aggregate data for specific customers by applying different statistical functions resulting in features quantifying users' behaviour.

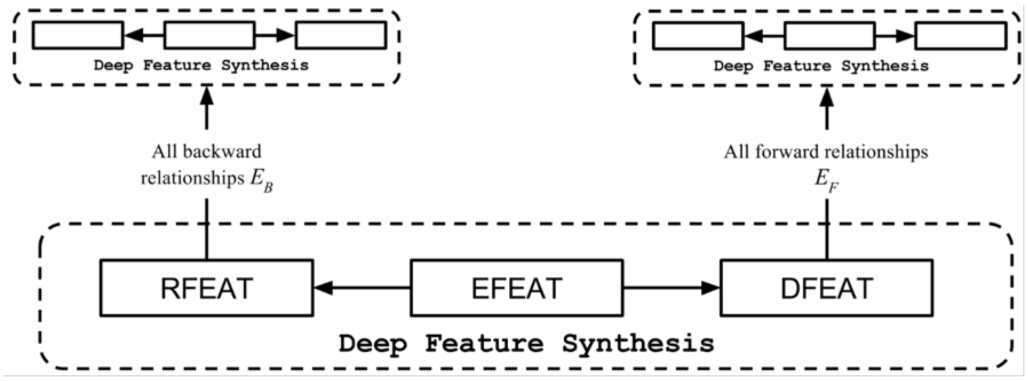


Figure 4. Deep Feature Synthesis [53].

Several features could be chosen to modify DFS's output. We selected all compatible features from the available collection of DFS features [54]. After choosing the features, we create entities from each file, thereafter defining the relationships between the entities. Figure 5 represents the relations between the files. The arrows represent many-to-one relationships. By applying a second primitive to the output of the first, DFS explores a vast space of meaningful features, resulting in 1000 user-specific profiles with 69,738 features in total.

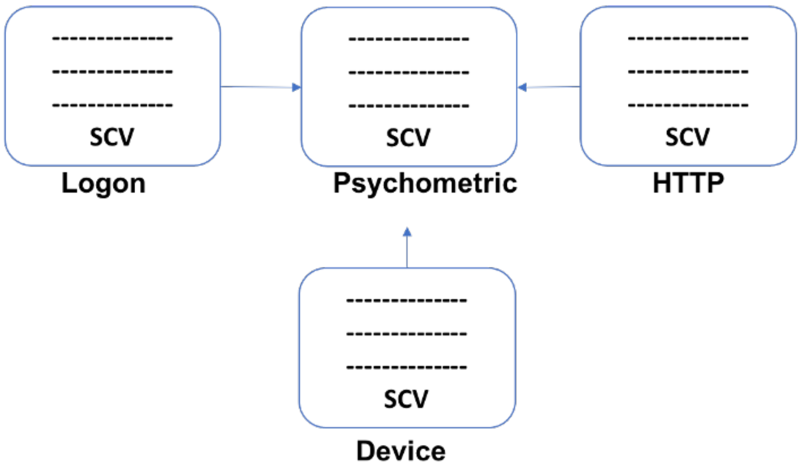


Figure 5. Illustration of the relations between the files.

3.3.2. PCA

PCA is a statistical process that decomposes a multivariate dataset into principal components using an orthogonal transformation. The original data's variance is preserved as much as possible by the PCA. Given that the principal components are orthogonal to the primary components and that the first component explains the majority of the data variability, each principal component has the greatest amount of variation that may be accommodated. It produces a condensed collection of the most important uncorrelated characteristics, which are linear combinations of the initial features [55].

In the experiments, the datasets are pre-processed by normalising each column in the original features set to have zero mean and unit variance. The most important uncorrelated features, which are linear combinations of the initial set of characteristics, were then obtained by using PCA with 95% of the variance.

3.3.3. SMOTE

Problems involving imbalanced datasets, in which instances of one class dominate instances of the other class, are frequently encountered in real-world applications. Without a particular countermeasure for imbalanced training data, the resulting decision boundary will bias classification performance for the majority class; however, the classification performance for the positive class degrades. There are two techniques for dealing with the problem of an imbalanced dataset: resampling methods (resulting in a balanced dataset) and imbalanced learning algorithms (anomaly detection methods, such as OCSVM and isolation forest).

Due to the imbalanced nature of the CERT dataset, where we have only 30 positive instances out of 1000. Thus, we used these two distinct methods to obtain satisfactory performance. In this study, SMOTE was not used for all models; rather, it was only used for classification models. CERT is an imbalanced dataset, as the anomalous samples are significantly less than the standard samples. We use SMOTE to balance the dataset and adjust the class distribution to get a good performance in the minority class.

3.4. Insider Threat Models

Due to the imbalanced nature of the CERT dataset, we used two different types of machine learning approaches: classification methods and anomaly detection methods. The models that were employed are described in detail in the following sections.

3.4.1. Anomaly Detection Models

The method employed to deal with the insider threat problem is determined by data availability. Previous insider attacks in the organisation would consist of only normal instances rather than a mixture of normal and abnormal instances. This section will look at cases where an organisation's data availability maturity is medium (i.e., only normal behaviour data are available).

The suggested framework detects harmful insider threats using one-class support vector machine (OCSVM) and isolation Forest (iForest) as base approaches. Because the data are highly skewed (30:1000), we want to make a model that treats the 30 instances as anomalies or outliers. The basic aim of OCSVM and iForest training is to identify a function, f, that returns a positive (+) result when applied to a point within f and a negative ($-$) result otherwise.

We take 70% of the typical data for the training phase after labelling it to $(-1, 1)$ to make it appropriate to the output of the OCSVM and iForest. We add the remaining 30% to the threat instances for the testing phase. For OCSVM, we use gamma = 0.0001 and nu = 0.001 with RBF kernel, and for iForest, we used the following: number of estimators = 100, maximum number samples = 256, and contamination = 0.5.

3.4.2. Classification Models

When both normal and abnormal data are available, the system can learn from both data classes. A binary classification approach is used to address the insider threat detection problem. A total of four supervised classification models are tested: SVM, Random Forest (RF), Neural Network (NN), and AdaBoost, each of which is evaluated separately.

We used stratified k-fold cross-validation, which folds the data while maintaining the percentage of samples for each class. Furthermore, we use the Grid-Search optimisation algorithm [56] to maximise the ROC-AUC in order to have the best set of classifier hyperparameters. Since we are using cross-validation, a pipeline was used to avoid information leakage. For every fold, we used PCA and StandardScaler. PCA is used for dimensionality reduction while preserving 95% of the variance. StandardScaler standardises features by eliminating the mean and scaling to unit variance.

Preparation for training the SVM classifier begins with tuning the C, gamma, and kernel hyperparameters via the GridSearch technique. Gamma is the parameter of a Gaussian Kernel (which is used to handle non-linear classification), and C is the parameter that governs the cost of misclassification of the training data. A kernel translates the data to a higher dimension, allowing the classes to be separated. The parameter C presented for a range of C = {0.001, 0.05, 0.1, 1, 1.5, 10, 100, 1000}, gamma for a range of gamma = {0.01, 0.1}, and kernel for a range of r = {0.3, 0.4, 0.5, 0.6, 0.7}, kernel = {RBF, 'poly', 'gamma'}.

The experiment's neural network design comprises an input layer, two fully connected layers, and an output layer. The majority of the learnable parameters are usually included in the fully connected layers. Before the fully connected layer, we employed Dropout to prevent over-fitting and enhance the model generalisation. Dropout provides regularisation by removing a fraction of the previous layer's outputs, forcing the network to not over-rely on any particular input. We used batch normalisation with a ratio of 0.5 to accelerate training convergence and increase overall performance. It is accomplished by normalising each feature at the batch level during training. Figure 6 shows the model architecture.

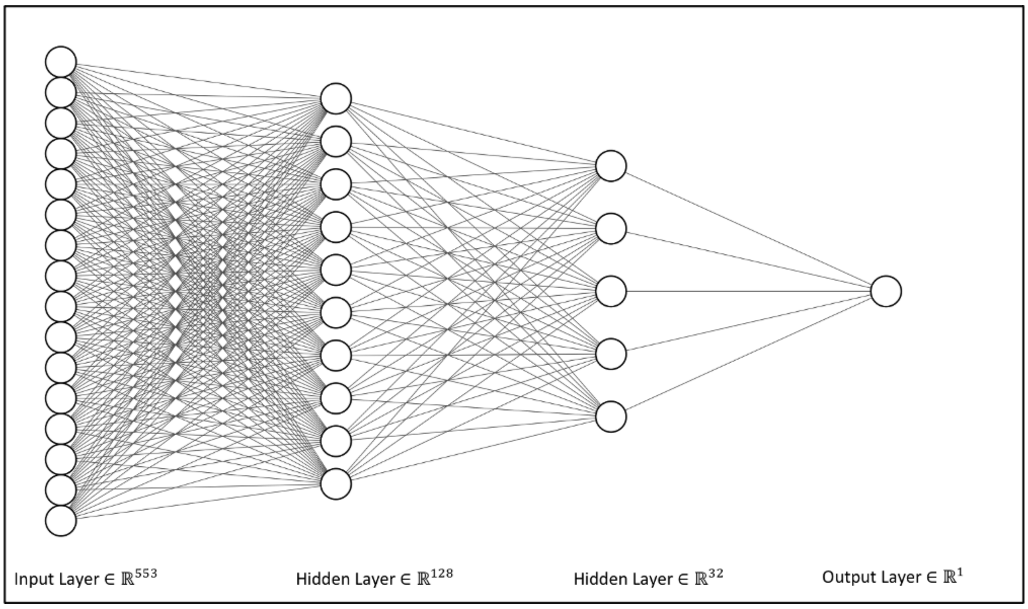

Input Layer $\in \mathbb{R}^{553}$
Hidden Layer $\in \mathbb{R}^{128}$
Hidden Layer $\in \mathbb{R}^{32}$
Output Layer $\in \mathbb{R}^{1}$

Figure 6. The proposed NN architecture.

In AdaBoost, we used a decision tree stump as a base estimator. Using a strong classifier might lead to an overfitting problem. The number of the base estimator is a significant hyperparameter that affects the classification results. We tuned the number of base estimators with the range {100, 500, 600, 700}. The learning rate in AdaBoost reflects how much each model contributes to the weights, and we tuned it using the range of {0.1, 0.5, 1}. The number of estimators in the random forest was tuned by {10, 20, 40, 50, 100, 150, 200, 500, 600}, as well as the number of features to take into account while looking for the best split = {'auto', 'sqrt', 'log2'}. We utilised entropy as a criterion, which is a function that measures the quality of a split in each estimator.

3.5. Evaluation

Following the prediction phase, we must use various evaluation metrics to assess the correctness of the model results, such as accuracy, precision, recall, and $\mathrm{F1-Score}$. The accuracy metric is a type of evaluation statistic that evaluates how accurate a classifier is. We simply add up the samples that were correctly predicted (true positive and true negative) and divide that amount by the number of samples to determine the accuracy using the confusion matrix; see Equation (1). In our case, we will classify the data into two categories: normal and abnormal, and the accuracy metric will give the percentage of the user's activities that are classified correctly.

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

where:

- TP: predicted abnormal activity is an abnormal activity;
- FP: predicted normal activity is an abnormal activity;
- FN: predicted abnormal activity is a normal activity;
- TN: predicted normal activity is a normal activity.

We can use these elements as input to calculate additional evaluation metrics, as demonstrated in Equation (2). Precision is a measurement of precision that provides us with a measure of exactness, which determines the number of all true predictions of anomaly (or abnormal activity) on all predictions. We can obtain 100% accurate predictions if the precision value is close to 1, which indicates that FP ≈ 0.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

The recall metric presented in Equation (3) provides an indicator to measure completeness. Thus, we will calculate the fraction of true predictions of abnormal activities out of the actual abnormal activities. If recall equals 1, it means we achieve high accuracy and FN = 0. Precision and recall both provide a metric for determining the efficacy of the proposed models in anomaly detection.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

The $\text{F1} - \text{Score}$, which is shown in Equation (4) and reflects the accuracy of the models' test set, will be used to determine the harmony between precision and recall measurements. We will reach the best value if the $\text{F1} - \text{Score} = 1$ and the worst value if the $\text{F1} - \text{Score} = 0$.

$$\text{F1} - \text{Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4}$$

4. Experimental Results

Insider threat detection has been the subject of extensive research for many decades. The literature reveals a wide range of solution approaches used in this issue, making method comparisons in terms of data, strategies, and aims challenging. We attempted to incorporate various contexts in which insider detection has been explored based on previous research. Our results improve previous results with the help of the feature engineering phase, including DFS. In this section, we present anomaly detection methods in the first section and classification methods in the second.

4.1. Anomaly Detection

Anomaly detection is the process of detecting deviations from normal behaviour in the data. We used two distinct anomaly detection methods, one-class SVM and isolation Forest. We train the models on the negative class, i.e., the normal class, which represents most users. We test the models' efficiency using the positive class with a subset of the negative class.

Recall that our dataset contains 1000 users, 30 of whom are abnormal or positive class. We divide the dataset into 80% and 20% for training and testing, respectively. The 80% for training contains all negative class users. We used 20% of the training set as a validation set. The 20% for testing has all 30 users as well as the remaining negative class. Table 5 shows the results on the CERT4.2 dataset after employing DFS and PCA on OCSVM and isolation Forest.

Table 5. Results of the anomaly detection methods.

Model	Precision	Recall	F1 − Score	Accuracy
OCSVM	0.94	0.86	0.89	0.86
iForest	0.92	0.91	0.88	0.91

4.2. Classification

Finding a suitable classifier that can identify the class of a testing instance based on the provided features related to that instance is the main goal of the classification task. We used four ML classification models to classify the users into normal and abnormal: NN, SVM, AdaBoost, and random forest. We aimed to reduce the imbalanced dataset problem and avoid the ML models' bias; we used the SMOTE balancing technique only in the training set to ensure no data leakage and received a balanced dataset of 1552 instances. Table 6 shows the results of supervised machine learning algorithms after DFS, PCA, and SMOTE on the unseen test data part of CERT r4.2. All experiments were performed using the grid search optimisation algorithm with 10-fold cross-validation. Additionally, Table 7 displays the results without utilising any balancing techniques.

Table 6. Results of classification methods with SMOTE.

Model	Precision	Recall	F1 − Score	Accuracy
NN	0.98	0.95	0.96	0.95
SVM	1.00	1.00	1.00	1.00
AdaBoost	0.98	0.94	0.95	0.94
RandomForest	0.95	0.72	0.81	0.72

Table 7. Results of classification methods.

Model	Precision	Recall	F1 − Score	Accuracy
NN	0.94	0.97	0.96	0.97
SVM	1.00	1.00	1.00	1.00
AdaBoost	0.99	0.98	0.98	0.98
RandomForest	0.94	0.97	0.96	0.97

4.3. Discussion

There were four distinct classification models that we utilised. Compared to the anomaly detection methods, the ML classification models produced the best overall results. The SVM model achieved the best results by successfully identifying all threat users. The primary step in the SVM experiment is finding the best combination of C and gamma parameters. A high value of C attempts to minimise the misclassification of the training data, while a low value smooths the model. Conversely, if the gamma value is too large, it will lead to an overfitting problem.

As the findings show, relying on SMOTE to achieve an equal balance with the majority class may not always the best option, depending on the desired outcome. Despite the fact that SMOTE did not raise the $F1 - score$ in the NN model, it boosted recall at the expense of accuracy and precision, an entirely desirable outcome when it comes to detecting insider threats, as a few false alarms are more desirable than undetected attacks. We would also like to emphasise that the original SMOTE paper [52] achieved excellent results by combining SMOTE and random under-sampling. Under-sampling was not used in this study because all data instances were necessary. Random forest and AdaBoost are machine learning algorithms designed explicitly for imbalanced datasets. We note that SMOTE is not required to achieve excellent results with these two methods. The overall performance of AdaBoost is superior to that of the random forest in all cases.

A comparison of other previous similar methodologies is presented in Table 8. The main similarity is that they handle collecting the data the same way as this research, where each data instance represents a user feature vector. The SVM model obtained the highest overall performance compared to the other algorithms. It achieved 100% accuracy. Furthermore, it has a higher recall (100%) and fewer false alarms. Additionally, the CNN model [34] has achieved a high accuracy (100%). However, since no values are provided for the other evaluation metrics, it is difficult to ascertain how well they are doing.

Table 8. Comparison with different classification methods.

Model	Accuracy	F1 − Score	Recall	Precision
CNN [29]	100%	n/a	n/a	n/a
Bio-Inspired models [57]	n/a	100%	n/a	70%
Autoencoder [58]	92%	96%	96%	94%
DBN-OCSVM [39]	87.79%	n/a	n/a	n/a
DNN [16]	96%	95%	n/a	n/a
Generative adversarial network [17]	n/a	n/a	76%	n/a
Light Gradient Boosting [40]	99.47%	92.26%	n/a	0.83%
CNN [29]	99%	99.3%	99.32%	99.29%
Random forest [59]	98%	n/a	n/a	99.32%
Our method	100%	100%	100%	100%

5. Conclusions

This study utilised machine learning techniques to detect malicious activity. A system pipeline that is able to accurately detect malicious users as well as being resilient to different organisational structures and updates to the log file structure was proposed. We used automated deep feature engineering to improve the detection of malicious insider users. DFS results in a vast number of meaningful features that hasten the feature engineering phase and help non-expert domains achieve results with good descriptive features. We examined the influence of the SMOTE oversampling technique to reduce the impact of data with an uneven class distribution because CERT is an extremely unbalanced dataset. The results showed that SMOTE helps increase recall over precision.

To increase system performance, we used PCA to minimise the number of features and reduce redundancy. To classify behaviours as malicious or normal, we tested several ML algorithms, either through anomaly detection methods or classification ones. The results of the experiments indicate that the SVM classification method outperforms the other models. The algorithm was trained and tested using a dataset of 1000 created feature vectors based on user action sequences, including both malicious and regular activity. We conclude that with the methodology used, the malicious users of the information system were effectively detected. The forecasting of the results was successful, with a percentage that reached 100%.

However, the study is not without some drawbacks. Although the system is designed to be resilient in the face of log file structural changes, it still requires a retraining phase; to build a different set of features that correspond to the new structure. The time required should be taken into consideration by the security officer. In addition, enrolling new users and learning their specific behavioural patterns necessitates collecting some usage pattern data first, i.e., the system is unable to detect malicious users who happen to be newcomers to the organisation. Finally, the system will require periodic retraining in order to keep the user baselines up to date.

Future work may consider different scenarios from the CERT insider threat dataset. In addition, organisations should be targeted, and a data-sharing agreement should be sought to attain access to real-world insider threat data and assess the methodologies described in this study. Future work should also consider using DFS in other problem domains.

Author Contributions: Conceptualisation, N.A.; methodology, N.A. and B.B.S.; software, B.B.S.; validation, N.A. and B.B.S.; formal analysis, N.A. and B.B.S.; resources, B.B.S.; data curation, B.B.S.; writing—original draft preparation, N.A. and B.B.S.; writing—review and editing, N.A. and B.B.S.; visualisation, B.B.S.; supervision, N.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in the experiment is available at https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099 (accessed on 27 November 2022).

Acknowledgments: This research project was supported by a grant from the "Research Center of the Female Scientific and Medical Colleges", Deanship of Scientific Research, King Saud University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Greatest Threat. Available online: https://www.darkreading.com/vulnerabilities---threats/greatest-threat/d/d-id/1269416 (accessed on 26 December 2022).
2. Noever, D. Classifier Suites for Insider Threat Detection. *arXiv* **2019**, arXiv: 1901.10948.
3. Cappelli, D.; Moore, A.; Trzeciak, R. *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*; Addison-Wesley: Boston, MA, USA, 2012.
4. Cheng, L.; Liu, F.; Yao, D. Enterprise data breach: Causes, challenges, prevention, and future directions. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2017**, *7*, e1211. [CrossRef]
5. Kulik, T.; Dongol, B.; Larsen, P.G.; Macedo, H.D.; Schneider, S.; Tran-Jørgensen, P.W.; Woodcock, J. A survey of practical formal methods for security. *Form. Asp. Comput.* **2022**, *34*, 1–39. [CrossRef]
6. Rauf, U.; Shehab, M.; Qamar, N.; Sameen, S. Formal approach to thwart against insider attacks: A bio-inspired auto-resilient policy regulation framework. *Future Gener. Comput. Syst.* **2021**, *117*, 412–425. [CrossRef]
7. Krichen, M.; Lahami, M.; Cheikhrouhou, O.; Alroobaea, R.; Maâlej, A.J. Security testing of internet of things for smart city applications: A formal approach. In *Smart Infrastructure and Applications*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 629–653.
8. Krichen, M.; Mihoub, A.; Alzahrani, M.Y.; Adoni, W.Y.H.; Nahhal, T. Are Formal Methods Applicable To Machine Learning And Artificial Intelligence? In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 23–26 May 2022; pp. 48–53. [CrossRef]
9. Larsen, K.; Legay, A.; Nolte, G.; Schlüter, M.; Stoelinga, M.; Steffen, B. Formal Methods Meet Machine Learning (F3ML). In Proceedings of the Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning, Rhodes, Greece, 22–30 October 2022; Margaria, T., Steffen, B., Eds.; Springer Nature Switzerland: Cham, Switzerland, 2022; pp. 393–405.
10. Urban, C.; Miné, A. A review of formal methods applied to machine learning. *arXiv* **2021**, arXiv:2104.02466.
11. Chen, H.; Zhang, H.; Si, S.; Li, Y.; Boning, D.; Hsieh, C.J. Robustness verification of tree-based models. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
12. Ranzato, F.; Zanella, M. Robustness verification of support vector machines. In Proceedings of the International Static Analysis Symposium, Porto, Portugal, 8–11 October 2019; Springer: Cham, Switzerland, 2019; pp. 271–295.
13. Jang, M.; Ryu, Y.; Kim, J.S.; Cho, M. Against Insider Threats with Hybrid Anomaly Detection with Local-Feature Autoencoder and Global Statistics (LAGS). *IEICE Trans. Inf. Syst.* **2020**, *E103.D*, 888–891. [CrossRef]
14. Kim, T.Y.; Cho, S.B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **2018**, *106*, 66–76. [CrossRef]
15. Tuor, A.; Kaplan, S.; Hutchinson, B.; Nichols, N.; Robinson, S. Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams. *arXiv* **2017**, arXiv: 1710.00811.
16. Karev, D.; McCubbin, C.; Vaulin, R. Cyber Threat Hunting Through the Use of an Isolation Forest. In Proceedings of the 18th International Conference on Computer Systems and Technologies, Ruse, Bulgaria, 23–24 June 2017; Association for Computing Machinery: New York, NY, USA, 2017; CompSysTech'17, pp. 163–170. [CrossRef]
17. Gavai, G.; Sricharan, K.; Gunning, D.; Hanley, J.; Singhal, M.; Rolleston, R. Supervised and Unsupervised methods to detect Insider Threat from Enterprise Social and Online Activity Data. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* **2015**, *6*, 47–63. [CrossRef]
18. Lin, L.; Zhong, S.; Jia, C.; Chen, K. Insider Threat Detection Based on Deep Belief Network Feature Representation. In Proceedings of the 2017 International Conference on Green Informatics (ICGI), Fuzhou, China, 15–17 August 2017; pp. 54–59. [CrossRef]
19. Meng, F.; Lou, F.; Fu, Y.; Tian, Z. Deep Learning Based Attribute Classification Insider Threat Detection for Data Security. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; pp. 576–581. [CrossRef]
20. Kim, T.; Park, N.K.; Cho, H.; Kang, P. Insider Threat Detection Based on User Behavior Modeling and Anomaly Detection Algorithms. *Appl. Sci.* **2019**, *9*, 4018. [CrossRef]
21. Sharma, B.; Pokharel, P.; Joshi, B. User Behavior Analytics for Anomaly Detection Using LSTM Autoencoder–Insider Threat Detection. In Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok, Thailand, 1–3 July 2020; ACM: Bangkok, Thailand, 2020; pp. 1–9. [CrossRef]
22. Orizio, R.; Vuppala, S.; Basagiannis, S.; Provan, G. Towards an Explainable Approach for Insider Threat Detection: Constraint Network Learning. In Proceedings of the 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), San Antonio, TX, USA, 5–7 September 2022; pp. 42–49. [CrossRef]
23. Tian, Z.; Shi, W.; Tan, Z.; Qiu, J.; Sun, Y.; Jiang, F. Deep Learning and Dempster-Shafer Theory Based Insider Threat Detection. *Mob. Netw. Appl.* **2020**. [CrossRef]

24. Haidar, D.; Gaber, M.M. Data Stream Clustering for Real-Time Anomaly Detection: An Application to Insider Threats. In *Clustering Methods for Big Data Analytics: Techniques, Toolboxes and Applications*; Nasraoui, O., Ben N'Cir, C.E., Eds.; Unsupervised and Semi-Supervised Learning; Springer International Publishing: Cham, Switzerland, 2019; pp. 115–144. [CrossRef]
25. Yuan, F.; Cao, Y.; Shang, Y.; Liu, Y.; Tan, J.; Fang, B. Insider Threat Detection with Deep Neural Network. In Proceedings of the ICCS, Wuxi, China, 11–13 June 2018. [CrossRef]
26. Raval, M.S.; Gandhi, R.; Chaudhary, S. Insider Threat Detection: Machine Learning Way. In *Versatile Cybersecurity*; Conti, M., Somani, G., Poovendran, R., Eds.; Advances in Information Security; Springer International Publishing: Cham, Switzerland, 2018; pp. 19–53. [CrossRef]
27. Malhotra, P.; Vig, L.; Shroff, G.M.; Agarwal, P. Long Short Term Memory Networks for Anomaly Detection in Time Series. In Proceedings of the ESANN, Bruges, Belgium, 22–23 April 2015.
28. Kwon, D.; Natarajan, K.; Suh, S.C.; Kim, H.; Kim, J. An Empirical Study on Network Anomaly Detection Using Convolutional Neural Networks. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; pp. 1595–1598. ISSN 2575-8411. [CrossRef]
29. Koutsouvelis, V.; Shiaeles, S.; Ghita, B.; Bendiab, G. Detection of Insider Threats using Artificial Intelligence and Visualisation. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 437–443. [CrossRef]
30. Sheykhkanloo, N.M.; Hall, A. Insider Threat Detection Using Supervised Machine Learning Algorithms on an Extremely Imbalanced Dataset. *Int. J. Cyber Warf. Terror.* **2020**, *10*, 1–26. [CrossRef]
31. Singh, M.; Mehtre, B.M.; Sangeetha, S. User Behavior Profiling using Ensemble Approach for Insider Threat Detection. In Proceedings of the 2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA), Hyderabad, India, 22–24 January 2019; pp. 1–8. ISSN 2640-0790. [CrossRef]
32. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48. [CrossRef]
33. Ren, Y.; Wu, Y. Convolutional deep belief networks for feature extraction of EEG signal. In Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 6–11 July 2014; pp. 2850–2853. ISSN 2161-4407. [CrossRef]
34. Al-Mhiqani, M.N.; Ahmad, R.; Zainal Abidin, Z. An Integrated Imbalanced Learning and Deep Neural Network Model for Insider Threat Detection. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 2021. [CrossRef]
35. Gayathri, R.G.; Sajjanhar, A.; Xiang, Y.; Ma, X. Multi-class Classification Based Anomaly Detection of Insider Activities. *arXiv* **2021**, arXiv: 2102.07277.
36. Mohammed, M.A.; Kadhem, S.M.; Maisa'a, A.A. Insider Attacker Detection Using Light Gradient Boosting Machine. *Tech-Knowledge* **2021**, *1*, 48–66.
37. Singh, M.; Mehtre, B.M.; Sangeetha, S. Insider Threat Detection Based on User Behaviour Analysis. In Proceedings of the Machine Learning, Image Processing, Network Security and Data Sciences, Silchar, India, 30–31 July 2020; Bhattacharjee, A., Borgohain, S.K., Soni, B., Verma, G., Gao, X.Z., Eds.; Communications in Computer and Information Science; Springer: Singapore, 2020; pp. 559–574. [CrossRef]
38. Rastogi, N.; Ma, Q. DANTE: Predicting Insider Threat using LSTM on system logs. *arXiv* **2021**, arXiv: 2102.05600.
39. Gayathri, G.R.; Sajjanhar, A.; Xiang, Y. Image-Based Feature Representation for Insider Threat Classification. *arXiv* **2019**, arXiv: 1911.05879.
40. Aldairi, M.; Karimi, L.; Joshi, J. A Trust Aware Unsupervised Learning Approach for Insider Threat Detection. In Proceedings of the 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA, July 30–1 August 2019; pp. 89–98. [CrossRef]
41. Kim, D.W.; Hong, S.S.; Han, M.M. A study on Classification of Insider threat using Markov Chain Model. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 1887–1898.
42. Le, D.C.; Nur Zincir-Heywood, A. Machine learning based Insider Threat Modelling and Detection. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Washington, DC, USA, 8–12 April 2019; pp. 1–6. ISSN 1573-0077.
43. Al-Mhiqani, M.N.; Ahmad, R.; Zainal Abidin, Z.; Yassin, W.; Hassan, A.; Abdulkareem, K.H.; Ali, N.S.; Yunos, Z. A Review of Insider Threat Detection: Classification, Machine Learning Techniques, Datasets, Open Challenges, and Recommendations. *Appl. Sci.* **2020**, *10*, 5208.
44. Lo, O.; Buchanan, W.J.; Griffiths, P.; Macfarlane, R. Distance Measurement Methods for Improved Insider Threat Detection. *Secur. Commun. Netw.* **2018**, *2018*, e5906368.
45. Yuan, S.; Wu, X. Deep Learning for Insider Threat Detection: Review, Challenges and Opportunities. *arXiv* **2020**, arXiv:2005.12433.
46. Hermans, M.; Schrauwen, B. Training and Analysing Deep Recurrent Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013; Curran Associates, Inc.: Red Hook, NY, USA, 2013, Volume 26.
47. Wang, G.; Hao, J.; Ma, J.; Huang, L. A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. *Expert Syst. Appl.* **2010**, *37*, 6225–6232. [CrossRef]

48. Anomaly Detection at Multiple Scales. Available online: https://www.darpa.mil/program/anomaly-detection-at-multiple-scales (accessed on 28 March 2022).
49. Statistical Methods for Computer Intrusion Detection. Available online: http://www.schonlau.net/intrusion.html (accessed on 28 March 2022).
50. *Insider Threat Test Dataset*; Software Engineering Institute: Pittsburgh, PA, USA, 2016.
51. Glasser, J.; Lindauer, B. Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data. In Proceedings of the 2013 IEEE Security and Privacy Workshops, San Francisco, CA, USA, 23–24 May 2013; pp. 98–104. [CrossRef]
52. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
53. Kanter, J.M.; Veeramachaneni, K. Deep feature synthesis: Towards automating data science endeavors. In Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–21 October 2015; pp. 1–10. [CrossRef]
54. Primitives | Featuretools. Available online: https://primitives.featurelabs.com/ (accessed on 7 November 2022).
55. Principal Component Analysis for Special Types of Data. In *Principal Component Analysis*; Jolliffe, I.T., Ed.; Springer: New York, NY, USA, 2002; pp. 338–372. ._13. [CrossRef]
56. 3.2. Tuning the Hyper-Parameters of an Estimator. Available online: https://scikit-learn.org/stable/modules/grid_search.html (accessed on 17 May 2022).
57. Nicolaou, A.; Shiaeles, S.; Savage, N. Mitigating Insider Threats Using Bio-Inspired Models. *Appl. Sci.* **2020**, *10*, 5046.
58. Pantelidis, E.; Bendiab, G.; Shiaeles, S.; Kolokotronis, N. Insider Threat Detection using Deep Autoencoder and Variational Autoencoder Neural Networks. In Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 26–28 July 2021; pp. 129–134. [CrossRef]
59. Le, D.C.; Zincir-Heywood, N. Exploring anomalous behaviour detection and classification for insider threat identification. *Int. J. Netw. Manag.* **2021**, *31*, e2109.