Insider Threat Detection using Profiling and Cyber-persona Identification

Badis Racherache

A Thesis
in
The Concordia Institute
for
Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science (Information Systems Security) at
Concordia University
Montréal, Québec, Canada

May 2021

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Badis Racherache

Entitled: Insider Threat Detection using Profiling and Cyber-persona Identification

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ Examiner
Dr. Amr Youssef

_____ Examiner
Dr. Chadi Assi

_____Supervisor
Dr. Mourad Debbabi

Approved by _____
Dr. Mohammad Mannan, Graduate Program Director

May 2021 _____
Dr. Mourad Debbabi, Interim Dean Gina Cody
School of Engineering and Computer Science

Abstract

Insider Threat Detection using Profiling and Cyber-persona Identification

Badis Racherache

Nowadays, insider threats represent a significant concern for government and business organizations alike. Over the last couple of years, the number of insider threat incidents increased by 47%, while the associated cost increased by 31%[1]. In 2019, Desjardins, a Canadian bank, was a victim of a data breach caused by a malicious insider who exfiltrated confidential data of 4.2 million clients[2]. During the same year, Capital One was also a victim of a data breach caused by an insider who stole the data of approximately 140 thousand credit cards[3]. Thus, there is a pressing need for highly-effective and fully-automatic insider threat detection techniques to counter these rapidly increasing threats. Also, after detecting an insider threat security event, it is essential to get the full details on the entities causing it and to gain relevant insights into how to mitigate and prevent such events in the future. In this thesis, we propose an elaborated insider threat detection system leveraging user profiling and cyber-persona identification. We design and implement the system as a framework that employs a combination of supervised and unsupervised machine learning and deep learning techniques, which allow modelling the normal behaviour of the insiders passively by analyzing their network traffic. We can deploy the framework as part of online traffic monitoring solutions for insider profiling and cyber-persona identification as well as for detecting anomalous network behaviours. The different models employed are assessed using specific metrics such as *Accuracy*, *F1 score*, *Recall* and *Precision*. The conducted experimental evaluation indicates that the proposed framework is efficient, scalable, and suitable for near-real-time deployment scenarios.

[1] `https://www.proofpoint.com/`, accessed on November 15, 2020

[2] `https://www.cbc.ca/`, accessed on November 15, 2020

[3] `https://www.securityinfowatch.com/`, accessed on November 15, 2020

Acknowledgments

First of all, I would like to express my heartfelt gratitude to my supervisor, Professor Mourad Debbabi, for his wise guidance and continuous support throughout my graduate studies. Thank you for allowing me to grow not only academically but personally and professionally as well. I was fortunate enough to work under your supervision, for which I am very grateful.

I wish to express my utmost gratitude to all my colleagues who motivated me and created such a great environment to work in. The laboratory and my learning experience would not have been the same without you. I owe special thanks to my colleagues and great friends Abdullah Qasem, Paria Shirani, Andrei Soeanu, ElMouatez Karbab, Dhiaa Elhak Rebbah and Meriem Ferdjouni who provided me with a lot of help and encouragement whenever needed throughout this journey. Many thanks my friends Sabri Belkacemi, Abdallah Benzine, Razine Bouache and Houssem Bordjiba for their sincere concern and friendship. I want to thank you for giving me much encouragement every time.

Moreover, I would like to send special and warm thanks to my whole family. In particular, I must express special gratitude and appreciation to my uncle, Fawzi Drouiche, and his family. Words can hardly express how much I appreciate everything they have done for me during my master's studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Nowadays, businesses depend on Internet connectivity as part of their workflow activities. However, the wide adoption of information technology (IT) as part of the business process was accompanied by an exponentially increasing number of reported cyber-attacks[1]. An insider is a current or former legitimate user or employee who has privileges, including access to the organization's IT infrastructure [22, 41]. A persona refers to the insider's role within an organization [12]. However, recent studies outline that 65% of reported attacks against various organizations can be traced to malicious insiders[2]. Moreover, insider attacks incur a whopping cost of around $400 billion yearly, with $348 billion directly tied to privileged users[3]. Detecting insider threats represents a key concern for businesses and governmental organizations alike due to the potential impact and related cost of such cyber-attacks on their operations. Consequently, this topic is also of high interest in the research community. The cyber-attacks planned by insiders might be more critical and potentially more damaging than the attacks initiated by outside actors due to the following reasons:

[1] `https://www.cpomagazine.com/`, accessed on November 15, 2020
[2] `https://www.pwc.co.uk/`, accessed on November 15, 2020
[3] `https://www.sans.org`, accessed on November 15, 2020

1. Traditional systems such as intrusion detection systems (IDS) have no specific capabilities to deal with such situations.

2. Insiders already have access privileges compared to the outsiders.

Thus, based on the above mentioned facts, there is an urgent need to develop a system that models the network's insiders' normal behaviour. The system should also profile insiders, identify their cyber-persona types and flag any deviation from their normal behaviour.

1.2 Problem Statement

Insider threat attacks affect more than 34% of businesses around the globe every year [13]. Those attacks have increased by 47% since 2018[4]. In 2019, Desjardins, a Canadian bank, had a data breach incident caused by a malicious insider who stole the private information of 4.2 million clients[5]. Therefore, it is crucial to design and implement an insider threat detection framework based on the analysis of insiders' network behaviour and flag any deviation from their normal behaviour. The framework should have the capability to classify network traffic generated by insiders automatically, profile them, identify their cyber-persona types and flag any behavioural deviation. It should also handle specific situations, including an insider using different devices or a team sharing the same machine. This capability can be beneficial for security analysts to detect and mitigate insider threats before they cause damage to the company's infrastructure. Moreover, the system should prevent such threats from occurring in the future. Specifically, we aim at answering the following research questions:

1. How can we segregate network traffic generated from human user devices and traffic generated by machines?

[4] `https://www.pandasecurity.com`, accessed on November 15, 2020
[5] `https://www.cbc.ca/`, accessed on November 15, 2020

2. How can we model the insiders' behaviours?

3. How can we train models to identify insiders and their related cyber-persona types?

4. How can we use the trained models to flag insiders' deviations from their normal behaviour?

1.3 Objectives and Contributions

The objectives of this research are as follows:

- Design and implement a framework capable of segregating near-real-time network traffic generated from insiders' devices.
- Model the network behaviour of the insiders, profile them, identify their cyber-persona types and flag their behavioural deviations.
- Assess the suitability of multiple machine learning and deep learning techniques and conduct an extensive experimental evaluation and performance comparison.

The key contributions of the thesis are as follows:

1. We design and develop an automatic system to segregate network traffic generated by insiders and non-human users using machine learning techniques.

2. We propose a novel machine learning-based framework for insider profiling and cyber-persona identification based on network traffic.

3. We propose a user behaviour-based insider threat detection system.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Initially, Chapter 2 presents the background and the related work. Then, Chapter 3 provides an overview of our approach, the feature engineering process and the details of our methodology. Thereafter, Chapter 4 illustrates our experimental study and the results obtained using the different modules of our framework. Finally, Chapter 5, provides the concluding remarks and comments on future research directions.

Chapter 2

Background and Related work

In this chapter, we present the background and related work of this thesis. First, we define the concepts of user profiling, cyber-persona and insider threats in Section 2.1. Then, we present the related work, starting with user profiling, cyber-persona and insider threat detection in Section 2.2. Finally, we conclude this chapter in Section 2.3.

2.1 Background

This section describes the technical terms used in this thesis, namely user-profiling, cyber-persona, insider and insider threats. Then, we present the machine learning/deep learning techniques used in our framework. Finally, we present techniques we use to extract embeddings from our data.

2.1.1 User Profiling

A profile is a unique representation containing important information about a user. In users' context connected to a corporate network, a profile consists of a set of network traffic data and metadata generated from the users' devices [55]. Profiling users and modelling their

behaviour are used for advertising purposes by predicting the users' interests and intentions. Cybersecurity analysts also adopt the concept to detect users' malicious behaviour in corporate environments. Advanced user profiling involves applying machine learning techniques to model the users' network behaviour by analyzing their network traffic. In some situations, profiling may include the extraction of new specific user-related information from available and detailed user information [19]. A concrete example of applying user profiling for cybersecurity-related problems is provided by Shaman *et al.* [57] where the authors introduce a new user profiling system based on the analysis of network flows.

2.1.2 Cyber-Persona

In general, a persona is defined as the image that someone presents to the public or the role he/she plays in a specific context[1]. In our research context, a persona is defined as a specific role of insiders in a corporate environment performing the same tasks and duties [12]. Each type of personas represents particular job responsibilities or particular interests on a given topic. *Cyber-persona* can be defined as any persona type connected to the digital infrastructure of an organization[1]. In other words, a cyber-persona represents a group of insiders executing similar tasks (developers, programmers, consultants, etc.) and having similar behaviour on the network. For example, in Stathatos *et al.* [62], the authors identify users by applying graph analysis techniques on indirect features obtained from the metadata of the website forums that users are visiting.

2.1.3 Insider Threats

This section defines the insider's concept, the threats that an insider might cause, and their different types. We also present the different insider attacks and the challenges facing their detection:

[1] `https://www.business2community.com/`, accessed on November 15, 2020

Insider

An *insider* is a current or former legitimate user who has privileges, including access to the organization's IT infrastructure (e.g., business network, systems or data), with different privilege levels. This access generally requires direct interaction with one or many authentication mechanisms [22, 41].

Since the insider can be physically or logically present in the company, this term also covers any processes or software agents created and executed by a legitimate user [14]. The word "legitimate" represents the key difference between an insider and an outsider attacker. It means that the insider always has an entry point to the system and does not need to put a significant effort to gain more privileges compared to an outsider attacker.

According to Hunker *et al.* [25], there are many other scenarios in which we consider an entity as an insider:

- A former employee who has valid system credentials.
- An employee of a third-party development firm who developed its software and has information on accessing the system.
- An entity with physical access (e.g., janitor) who finds an unlocked machine.

Insider Threats

We define insider threats as any harmful acts causing damage to the organization's cybersecurity or data and initiated by someone inside the organization[2]. The insiders' dangerous actions can be illegitimate access to resources, abuse targeting the systems/computers, data exfiltration, data integrity loss, or any activities causing an interruption of business services. The insider threat is considered a daunting problem in many cybersecurity studies for many reasons [5, 14]:

[2] `https://www.clearswift.com`, accessed on November 15, 2020

- Most of the security tools do not handle insider threats.
- Insiders might have an extensive array of privileges.
- The insiders' in-depth knowledge of existing organizational policies and rules (unknown to an outsider attacker).

The leading cause of insider threats is insiders having elevated privileges required to fulfill their organization's tasks, and other insiders access sensitive data[3]. According to the literature, insider threats can be intentional or unintentional (e.g., due to negligence or lack of training). With respect to the unintentional cases, typical threats relate to situations where an insider visits a phishing web page or clicks on a malicious link received as part of an email. In contrast, intentional threats occur with premeditation on the insider part; it includes installing back-doors or unauthorized access to sensitive information and resources. In both scenarios, the threats can result in the sabotage of the IT systems, resource misuse, denial of service, and data loss or data exfiltration [14]. Such threats can have severe consequences on the IT infrastructure and business operations of an organization. Intentional insider threats can be influenced or motivated by the background of the insider, including[3]:

- Insider collaborating with direct competitors and intelligence agencies
- Insider seeking for a financial gain
- Insider member of political or social activism group
- Insider unhappy with his/her salary, supporting a new position with another company or starting his own business

It is also important to know the reasons behind the increase of insider threats during the past few years[3]:

[3] `https://advisory.kpmg.us`, accessed on November 15, 2020

- Workforce became more and more transitory and hence vulnerable to high turnover; therefore, employees became like free agents with less loyalty to their organizations.
- Incapacity of monitoring insiders working from home or using their own devices, this can also include third party contractors getting direct access to the organization's sensitive information without any monitoring.
- The growing use of the cloud puts more data outside of the company's immediate control.

Types of Insider Threats

According to the literature, we can categorize insider threats into three main categories: compromised, careless and malicious insiders[4].

- **Compromised Insiders.** This type is considered the most important threat because the insider is not aware that his system is compromised. It mainly occurs when an employee clicks on a phishing email link and grants access to the attacker.
- **Careless Insiders.** This type of insiders represents the main target of outside attackers due to the insiders' lack of security awareness; they could be insiders leaving their workstations un-locked, visiting suspicious websites or installing software from untrusted sources.
- **Malicious Insiders.** This type of insider might be a current or former insider having legitimate access to the corporate infrastructure. They can steal confidential data or any kind of intellectual property.

Types of Insider Attacks

Steven M. Bellovin *et al.* [4] classified the insider attacks into three types:

[4] `https://phoenixnap.com/`, accessed on November 15, 2020

- **Misuse of Access Privileges.** In this type of attack, an insider with malicious intentions takes advantage of his/her legitimate access privileges to execute malicious tasks.

- **Bypassing Defenses.** In this type of attack, the insider has an advantage over the outside attackers since they are already connected to the organization's system.

- **Access-control Failure.** In this type of attack, technical problems are the leading cause of the attack's success; in this case, the access system might not be configured correctly. Therefore, unauthorized insider gains access to the company's network.

Challenges of Insider Threat Detection

Even though insider threat detection is a hot topic, and many works have been done in this area, it still faces many challenges [38]:

- **Lack of Balanced Datasets.** Malicious events represent a small percentage of real-world datasets since they do not occur all the time, and they are mitigated quickly. Hence, it represents a significant challenge to applying machine learning and deep learning techniques for insider threat detection because those techniques require balanced datasets to be effective.

- **Less Focus on Temporal Features.** The temporal aspect of security-related events is as important as the other user's activity data. For example, insider copying files during work hours can be considered as usual. Still, if he copies the data at midnight or copies data during an extended period, it should be flagged as suspicious.

- **Attackers Adaptability.** Malicious insiders can always improve their attacks to evade the detection mechanisms, making the trained models inefficient against new attacks; hence, there is a need to develop adaptive detection systems.

Improving the Insider Threat Detection Capability

The following are some recommendations to improve the insider threat detection capability[5]:

- **Collect Insider Threats Data.** This technique involves selecting a set of insider threats, collecting data, and training detection models on it. The choice of insider threats depends on the organization's infrastructure.
- **Detect Spikes in Activity.** Monitor the network to detect any unusual spikes in the insiders' activity (e.g., it could be a high number of attempts to access sensitive data).
- **Monitor Access Attempts:** Monitor in real-time successful and failed access attempts, and flag the suspicious ones in terms of the number of attempts and their frequency during a given amount of time.
- **Restrict Access to Sensitive Data.** Prevent unauthorized insiders from accessing the organization's sensitive data using policies. Flag any continued access to files during a given amount of time and flag any insider trying to access different files.
- **Identify and Monitor Shared Accounts.** Monitor the use of those accounts is essential since they do not belong to a single insider. The most important parameter to consider is the frequency of logins.

2.1.4 Text Representation Techniques

This section will present the two text representation techniques we have used in our experimental analysis to represent the categorical features numerically. The first technique we have used is *FastText* [7], a text representation technique that represents each word as a bag of character n-grams; the summation of those n-grams represents the word representation.

[5] `https://blog.netwrix.com`. accessed on November 15, 2020

The main advantage of using this technique is the small amount of time required to train models on large corpora and its capability to compute a representation for unseen words. The second technique we have used is *paragraph2vector*; this technique is used to learn in an automatic way the embedding representation and the semantic in sequences of words from their appearances in a given context. Thus, the words appearing together will have a close representation in the embedding space [33].

2.1.5 Machine Learning and Deep Learning Techniques

Machine learning techniques are widely used to solve different problems, especially when classical algorithms become inefficient.

Unsupervised Machine Learning Techniques

Unsupervised machine learning techniques aim to find similar data groups with high similarity in the absence of labels. *K*-means [46], Auto-encoders [49] and Isolation Forests [36] are examples of clustering techniques used to solve various problems in the absence of labelled datasets.

- ***K*-Means.** *K*-means is one of the most used clustering algorithms. This algorithm creates K clusters (the number of clusters user-defined) by first finding the best K centroids, then each data point is assigned to the clusters closest to its centroid [46].

- **Autoencoders:** Autoencoders are a special architecture of neural networks used in an unsupervised way, with the particularity that the output is set as the same value of the features vector as the input. This algorithm's main idea is to learn an approximation to the identity functions using the back-propagation learning algorithm [49].

  Deep Autoencoders. This type of autoencoders has a high number of hidden layers; the number of hidden layers depends on the complexity of the problem being solved.

- **Isolation Forests.** This method was designed for anomaly detection purposes. An isolation forest is a set of isolation trees (an Isolation tree is built by randomly partitioning data until all the data points are separated or isolated). This technique is first built for the dataset being experimented with, then flag as anomalous data points having short average path lengths on the Isolation Trees. This method is known to have the number of trees that we desire to build and the sub-sampling size as the only two hyperparameters to configure to get the best performances [36].

Supervised Machine Learning Techniques

It is also called classification; it is applied on a labelled dataset and learns a function mapping between the data instances and their corresponding labels. Decision Trees, Random Forests [35] and neural networks are examples of such techniques.

- **Decision Tree.** A decision tree is tree-like graph-based decision support in a supervised way. Each node of the decision tree represents a test on the feature being analyzed; the test outcomes are branches leading to other nodes. Finally, a leaf represents the final prediction or the label [51].

- **Random Forest.** Random Forest is a powerful and widely used supervised machine learning algorithm for both classification and regression. This algorithm consists of an ensemble of decision trees; the number of decision trees within the ensemble defines the random forest model's robustness. Given a data instance, each decision tree outputs a class, and the class with the highest number of votes is considered the prediction of the random forest [9, 35].

- **Multi Layer Perceptron (MLP).** Multi-layer perceptron (MLP) is an artificial neural network mainly used for classification purposes. However, it can also be used for regression. Its design was inspired by the human brain and based on studies made by

cognitive scientists and neuroscientists. The perceptron is the main processing unit for MLPs; it receives inputs from the environment or other perceptrons. The perceptron's output results from the activation function's application on the weighted summation of its inputs. An MLP is composed of many layers of perceptrons. In the first layer, the number of perceptrons is the same as the number of features, the same thing for the output layer, in which the number of perceptrons is the same as the number of labels. It is determined empirically concerning the number of hidden layers and the number of perceptrons per hidden layer. Finally, the MLP is trained using the back-propagation algorithm [1].

- **Deep Learning (Supervised).** A deep neural network is a neural network with a high number of hidden layers depending on the problem being solved (the higher the number of hidden layers is, the deeper the model is). The main advantage of deep learning over classical machine learning techniques is that the classical machine learning techniques require feature engineering; in other words, the security expert should select the security event features. In contrast, in deep learning, no feature selection is needed. Moreover, deep learning helps solve more complex problems and can learn data representations at different abstractions levels. A deep neural network is composed of an input layer with a number of units equal to the number of features, an output layer with a number of units equals the number of classes and many hidden layers where the number of units per hidden layer depends on the neural network architecture, the nature of the data and the nature of the problem been solved. Each layer's units are connected to the next layer's units depending on the type of the model [34].

2.2 Related Work

In this section, we present the related work of this thesis. First, we present research initiatives dealing with user profiling and cyber-persona identification. Then, we present insider threat detection techniques.

2.2.1 User Profiling

Pang *et al.* [48] present a user identification system by analyzing the network flows generated by users connected to wireless networks, even if they are using pseudonyms. Conti *et al.* [11] present a system capable of identifying specific actions performed by android users when using a set of selected android applications. They apply advanced supervised machine learning techniques on the encrypted network flows generated by android devices and achieved 95% accuracy. Soh *et al.* [61] propose a novel framework for the early detection of insider threats by profiling insiders using aspect-based sentiment analysis and social network information. The authors profile the insiders' temporal sentiments using an ensemble of deep learning techniques. Shaman *et al.* [57] propose a novel user profiling system based on the analysis of features derived from application-level flow sessions. They have collected 60 days of network flow data generated by 23 users and use it to model the users' behaviour on the network using classification algorithms. They obtained up to 74% accuracy. Gratian *et al.* [20] investigate the usefulness of features extracted from network traffic to profile users and flag infected ones. They used two months of network traffic data collected from the university network; they applied different dimensionality reduction techniques to 36 features extracted from this data. Their detection capability uses a combination of supervised and unsupervised machine learning techniques. In their experiments, Principal Component Analysis [69] gave the best 10 features responsible for 92.8% of the users' variance, K-means [46] clustering algorithm segregated users into groups of benign and infected users. Finally, classification gave an accuracy equals to 79% and ROC AUC

up to 86%. Saltaformaggio *et al.* [54] present *NetScope*, a system capable of identifying fine-grained user activities through the passive analysis of encrypted network traffic. This last can learn different behavioural activities run by users using both Android and IOS devices. Their experimental evaluation used 35 popular application activities and obtained a precision equal to 78.04% and recall equals 76.04%.

2.2.2 Cyber-Persona Identification

Stathatos *et al.* [62] identify users on website forums. The authors employ features derived from website metadata extracted from websites visited by users. Similarity scores between users across various website forums are then derived using a decision tree and random forest. Experimental results show that their system achieves 96.3% of F_1 score. Yang *et al.* [17] use network access traces and network traffic data to identify cyber-persona types of users connected to the university WIFI internet access. They have collected two months of data generated from more than 2000 devices; each device's number of access records varies from 300 to 2000 access records; each record contains the device's mac address, the login logout time, and the location. The features engineering phase computes each connection's duration, the proportion of time spent in each department and uses under-sampling to solve the imbalanced dataset issue. Their system comprises three prediction components: implementing different classifiers (linear regression, Support Vector Machine (SVM), decision tree, etc.). The first two components use binary classifiers to identify whether a user is a student or a faculty member and whether they are an undergraduate or graduate. The third component detects the major of the student been identified. Their evaluation shows that in the first component (Faculty members and students segregation) and the second component(undergraduate from graduate), SVM gave the best accuracy with 87,67% and 88,77%, respectively. Finally, linear regression provides the best accuracy with 72,27% in the students' discipline prediction component. Bakhshi *et al.* [2] propose

a real-time network traffic-based campus users profiling system. They apply the K-means clustering algorithm to separate users into different clusters based on their behaviour. They obtained six unique types of users extracted from real-world data collected from a campus switch.

2.2.3 Insider Threat Detection

Insider threat detection can be classified as host-based (collecting and analyzing system calls, system logs, etc.) or network-based (analyzing network traffic). Since the last one is closely related to our work line, we will discuss the most prominent approaches in this area. According to the literature, the selected approaches can be classified based on the employed machine learning techniques as depicted in Fig 1. Unsupervised-based insider threat detection, approaches [18, 27, 37] mainly employ clustering algorithms such as *k*-means [46]. On the other hand, supervised-based techniques [16, 18, 30, 39] use classification algorithms such as random forests [35], linear regression and MLP. More recent approaches favour the use of deep learning techniques involving deep neural networks.

Detection based on Unsupervised Machine Learning

This type of approach use clustering algorithms such as *k*-means [46], isolation forests [36] and autoencoders [49]. Liu *et al.* [37] propose the use of auto-encoder [49] for insider detection. Their approach involves three components (log2corpus, feature extraction and insider threat detection). In the log2corpus component, the authors transform security-related logs into `Word2vec` [63] trainable corpus. The obtained corpus is unified and uses the same format, irrespective of the type of security-related event. The feature extraction component uses `Word2vec` feature vectors from log text, with the key benefit of not requiring any domain knowledge. The threat (malicious event) detection component employs an auto-encoder trained on the features generated by the feature extraction component. If the

Figure 1: Types of Insider Threat Detection: First Taxonomy

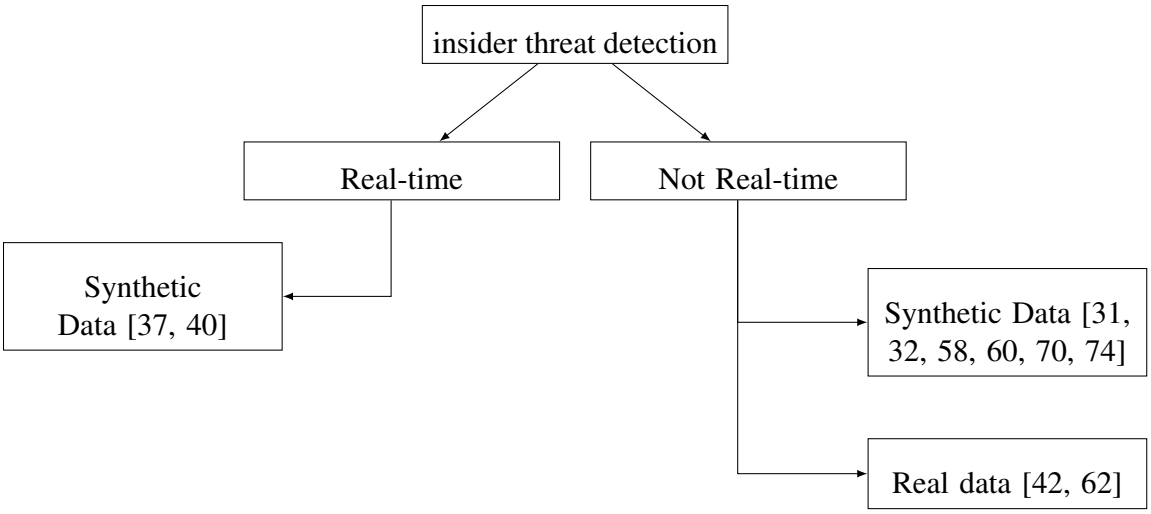


Figure 2: Types of Insider Threat Detection: Second Taxonomy

reconstruction error of the features is above a threshold, they flag the event as suspicious. Moreover, They flag any user associated with many suspicious events as a malicious insider. Gavai *et al.* [18] use social and online activity data in an enterprise to identify abnormal behaviour using isolation forests [36]. Kim *et al.* [27] propose a combination of behaviour modelling and anomaly detection algorithms to detect threats using three different types of user log data (daily activity summary, the topic contribution of e-mail and e-mail communication history per week) from the CERT insider threat test dataset [65]. For the anomaly detection, the authors use Gaussian density estimation [59], Principal Component Analysis (PCA) [75] and k-means [46] algorithms. Tuor *et al.* [67] propose a real-time anomalous network activity detection using a traffic log analysis-based system that leverages unsupervised deep learning models. They generate user-specific daily data vectors from the raw data taken from the CERT insider threat test dataset [65]. They feed the obtained vectors into the user's corresponding machine learning models to learn his/her normal behaviour and then flag abnormal behaviour and report it to the security practitioners. The authors employ Deep Neural Network (DNN), and Recurrent Neural Network (RNN) [44] as deep learning models. To compare the performance results of their system, they also employ other outlier detection methods, mainly one-class SVM (SVM) [64], Isolation Forests [36]

and PCA [69]. Zhang *et al.* [74] propose a new unsupervised, data-driven method using denoising autoencoders. The method is completely independent of any domain knowledge and insider's specific information. It is made up of three modules (data preprocessing, user behaviour construction and abnormal behaviour detection). In the data preprocessing module, they construct the insider's cross-domain behavioural descriptions using CERT dataset [65]. In their second module, they use denoising autoencoders to construct insider behaviour features. Finally, in the abnormal behaviour detection modules, they adopt the Gaussian Mixture Model (GMM) [47], Robust-Covariance and one-class SVM [43], Isolation Forest [36] and Local Outlier factor [10] to detect insider threats. Sharma *et al.* [58] propose an anomaly detection system by modelling users' behaviour using unsupervised machine learning techniques. First, they collect, aggregate, preprocess and normalize data collected from CERT insider threat test dataset [65]. Second, they model the user's normal behaviourby feeding normal data into LSTM RNN-based autoencoder. Finally, they use the autoencoder's reconstruction error as a metric to determine the maliciousness of the user behaviour; the higher is the reconstruction error more suspicious is the user's behaviour.

Detection based on Supervised Machine Learning

In this section, we discuss insider threat detection approaches employing supervised techniques, including traditional machine learning classification algorithms and deep learning algorithms. Yu *et al.* [71] employ mobile phone sensory data such as application usage and screen activity events to generate user profiles. They use those profiles to segregate users based on their gender, age and personality attributes in order to provide them with personalised services. User gender and age are estimated using random forest, while personality attributes are identified via Support Vector Regression (SVR). They experimentally evaluate the performance of their approach on real-world data. Duc *et al.* [30] leverage a

collection of machine learning algorithms, including logistic regression [28], random forest [35] and Artificial Neural Networks (ANN) for insider threat modelling and detection. They use data with multiple granularity levels to support the design of a user-centred system for insider threat detection. In the data collection and processing phases, they extract activity logs, organization structure and user information from CERT insider threat test dataset [65]. Subsequently, they generate two types of features: frequency features (eg. the number of copied files) and statistical features (eg. the size of an email). In the machine learning phase, they train logistic regression, random forests, and neural networks to detect unknown insiders based on limited ground truth. Finally, the security analyst generates feedback on all the detected insiders. This feedback is then used to validate the classification models. Pedro *et al.* [16] introduce an experimental study on the impact of feature normalization techniques and the exploration of temporal information on insider threat detection. In the second part of the previously mentioned work of Gavai *et al.* [18], the authors apply random forest [35] to detect insider threat activities using the indication when employees quit the company as labels. Lo *et al.* [39] introduce a Hidden Markov Model (HMM) based approach for insider threat detection using the CERT insider threat test dataset [65]. Subsequently, the authors conduct an experimental study on detecting the change of user behaviour by employing different distance metrics. Rashid *et al.* [52] present an insider threat detection system involving the modelling of the insider's normal behaviour using HMMs [56]. Their feature extraction module uses the CERT insider threat test dataset [65] raw data to generate sequences of user actions every week. In their Anomaly detection module, assuming that the first five weeks of data are clean (does not contain any abnormal behaviour), they train the HMM [56] on it. Then, they feed each sequence from the rest of the weeks to the model to get a corresponding prediction probability. The HMM model is then trained on that sequence. After going through all the sequences, they empirically select a threshold and use it as an indicator such that if

the prediction probability of a given sequence is below the threshold, an anomaly will be flagged. Yuan *et al.* [72] present a deep learning-based insider threat detection using the CERT insider threat test dataset [65]. Inspired by natural language processing, the authors consider the user's action as a word, his/her sequence of actions as a sentence, and they employ Long Short Term Memory (LSTM) [21] neural network to generate a representation of the user's "language". This representation is then fed to a Convolutional Neural Network (CNN) [29] model to detect anomalous behaviour. Sheykhkanloo *et al.* [60] conduct an experimental study on the impact of using a highly-imbalanced dataset on the performance of supervised machine learning techniques. Furthermore, they study the impact of data resampling on the performance results of three machine learning models: J48 decision tree, SVM [64], Naive Bayes (NB) [53] and random forests [9]. The authors experimentally prove that balancing the data does not improve the performance results of the employed models. However, it improves the model's training and testing time. Also, using parameters different from the default ones impacts the performance of the machine learning models in both scenarios (using imbalanced and balanced data), with a higher impact in the case of imbalanced data. Happa *et al.* [23] introduce an automated method to detect insider threat anomalies. In their experiments, they study the performance of applying the GMM while considering the use of contextual information to improve the performance of their method by taking advantage of security experts' feedback during the visual analysis of the data. They prove that their method can help to improve the detection rate and reduce the false alarm rate. Maloof *et al.* [42] present a machine learning-based system for detecting insider threats by flagging the insiders operating outside of their normal duties; this system is meant to help the cyber-security analysts in their investigations of the malicious insiders. In the first step of their system, they process network traffic and flag abnormal traffic based on security experts' presence. Second, they train a Bayesian Network (BN) to predict each event's threat score; if the score is above a given threshold, it will be flagged as suspicious.

Kandias *et al.* [26] propose a novel, interdisciplinary model combining computer science and psychology techniques to predict insider threats. They monitor the insiders' behaviour using the company's information systems, the insiders' psychological features to evaluate their aptitude to execute malicious acts. Le Duc *et al.* [32] present a machine learning-based and user-centred insider threat detection system composed of four modules: data collection, data pre-processing, machine learning and data analytic modules. In the data collection module, they collect user activity data, organization structure data and insider profiles information from the CERT insider threat test dataset [65]. The data pre-processing module aggregates and processes the data to generate vectors of features representing user activities; those features are either statistical or frequency features. The machine learning module trains and validates supervised machine learning models. Finally, the data analytic module displays instant alerts, user alerts, and malicious behaviour analysis to the network administrator. Elmrabit *et al.* [15] propose a novel approach for predicting insider threats related to data breaches. In their framework, they predict every insider's risk level and flag any potential threat risk. In the first phase of their framework, they collect three types of data from an educational institution and a small enterprise located in the United Kingdom. The three types of data are human factor, technological aspect and organizational data. They collect human factor data by addressing surveys to 70 insiders; they collect the technological aspect data by addressing surveys the It services while collecting organizational impact data by addressing a survey to the human resources department. For the small enterprise, the authors collect both the technological and the organizational data by addressing a survey to the director; however, the authors collect human factor data by addressing surveys to all the employees. After organizing, processing and exploring the data, they feed it to a Bayesian neural network. They compare the framework's obtained prediction results with the security expert's decisions. The comparison shows that the framework can achieve better predictions than security experts.

Jiuming Lu *et al.* [40] propose a deep learning-based insider threat detection system called *Insider Catcher*. They model the insiders' normal behaviour using system logs from the CERT insider threat test dataset [65], considering them as a structured natural sequence containing patterns that can be used as an indicator to detect malicious acts. In their experiments, they use LSTM as a deep learning technique, and for their comparison, they use PCA and SVM as classical anomaly detection models.

Detection Based on Other Techniques

Bishop *et al.* [6] propose an insider detection method; they consider each set of activities run by an insider within his/her specific task as a process and represent it formally using process modelling techniques. Also, they study how a process can be compromised and propose a solution to prevent it. Le Duc *et al.* [31] present a new hybrid user-centred abnormal behaviour and insider threat detection system combining supervised and unsupervised machine learning techniques and consider multiple data granularity levels on data collected from a CERT insider threat test dataset [65]. In the data pre-processing phase, they reduce the correlation between features and project the data into a low dimensional space using different data engineering techniques such as autoencoders [3], PCA [66], and random projection (RP) [68]. The unsupervised anomaly detection part of their system feeds unlabeled data to autoencoder. In contrast, in the supervised insider threat detection part, they feed the data to Linear Regression [45], ANN, random forest [9] and NB [53].

Table 1: Related Work Summary

References	Year	Unsupervised Machine Learning Techniques									Supervised Machine Learning Techniques													Other Techniques			Source of the Data	Nature Of the Data						
		Denoising Autoencoders	Gaussian Mixture	Robust Covariance	One-class Support Vector Machine	Isolation Forest	Local Outlier Factor	Auto-encoder	K-means	DBSCAN	LSTM RNN-based Autoencoder	Decision Trees	Random Forests	Linear Regression	Neural Networks	Xgboost Bayesian Neural Neutwork	Support Vector Machine	Naïve Byes	Logistic Regression	Convolutional Neural Network	LSTM	Multi Layer Perceptron	Deep Neural Network	Multi Fuzzy Classifier	Word2vec	Principal Component Analysis	Gaussian density estimation	Parzen window density estimation	Hidden Markov Model	Gaussian Mixture Model		Network Traffic	Sys Logs	Psychological Data	Organizational Information	User Information	Email Messages	Technological Aspect
Zhang *et al.* [74]	2020	●		●	●	●																								CERT	●	●					
Le Duc *et al.* [32]	2020												●	●	●															CERT	●	●			●	●	
Sheykhkanloo *et al.* [60]	2020											●	●					●	●											CERT	●						
Sharma *et al.* [58]	2020				●						●																			CERT	●	●					
Le Duc *et al.* [31]	2020							●					●		●					●	●					●					CERT	●	●					
Xiaoyun *et al.* [70]	2020									●																			●		CERT	●	●					
Elmrabit *et al.* [15]	2020						●																								Educational Institution and Small Business					●	●	
Liu *et al.* [37]	2019	●																							●					CERT	●	●					
Jiuming Lu *et al.* [40]	2019																	●				●								CERT	●	●					
Kim *et al.* [27]	2019								●																	●	●	●			CERT	●	●					
Yu *et al.* [71]	2019									●																			●		CERT	●	●					
Duc *et al.* [30]	2019												●		●						●									CERT	●	●					
Pedro *et al.* [16]	2019												●		●						●									CERT	●	●					
Soh *et al.* [61]	2019																													Enron Email Dataset						●	
Stathatos *et al.* [62]	2018											●	●																	Dark market and forums					●		
Lo *et al.* [39]	2018																				●								●	CERT	●	●					
Yuan *et al.* [72]	2018																					●	●							CERT	●	●					
Happa *et al.* [23]	2018																												●	CERT	●	●					
Tuor *et al.* [67]	2017				●	●																●	●		●	●					CERT	●	●					
Rashid *et al.* [52]	2016				●																								●		CERT	●	●					
Gavai *et al.* [18]	2015				●																									Vegas and CERT		●	●			●	
Maloof *et al.* [42]	2007																●													Operational corporate intranet	●			●			

Table 2: Comparison Between Different insider threat detection Research Papers

Reference	Performance Indicators(%)									Strengths	Weaknesses
	Acc.	F_1	**Rec.**	**Pre.**	**TP**	**FP**	**TN**	**Det.**	**AUC**		
Zhang *et al.* [74]	75	81	88.9	-	-	-	-	-	-	• Multiple clustering techniques • Comparison with outlier detection techniques	• Not Real-time • Accuracy is not high
Le Duc *et al.* [32]	-	77.6	-	72	-	1.4	-	86.5	-	• Multiple classifiers • Multiple data granularity Levels • Limited Ground truth • Different insider threats scenarios	• Not Real-time • Do not use temporal information
Sheykhk anloo *et al.* [60]	90	90	90	90	90	10	-	-	-	• Handle Imbalanced data • Detect 6 data Breach scenarios • Experimental Analysis of the importance of model's parameters	• Not Real-time • Detects few data breaches scenarios only
Sharma *et al.* [58]	90.2	-	91	-	-	9.8	90.2	-	0.95	• Detect unseen behaviour and flag anomalous patterns	• Not Real-time • No automatic feature extraction
Maloof *et al.* [42]	-	-	-	-	-	1.5	-	84	92	• Use real-world dataset	• Not Real-time • Detect 5 specific scenarios only
Le Duc *et al.* [31]	99.8	-	-	0.79	32	0.02	-	-	aa	• User centred detection • Multiple data granularity levels • Analysis performed on data instances, normal and malicious insiders	• Not Real-time
Xiaoyun *et al.* [70]	-	-	-	-	99	-	94	-	-	• Rapid detection of abnormal behaviour	• Detection depends on the amount of data • Do not handle new users and userswith less data
Liu *et al.* [37]	-	-	-	-	-	0.59	-	-	99.8	• Low false positive rate • Is real-time	• Need more evaluation metrics
Elmrabit *et al.* [15]	squared correlation = 0.87									• Combine technical, organizational and human factor related features • Framework's evaluation done on challenging conditions • Achieve better results than empirical judgment of security experts	• Probability distribution designed based on literature review
Jiuming Lu *et al.* [40]	-	80	90	72	-	-	-	-	-	• Is real-time	• Does not provide comparison with other models • Needs more evaluation metrics
Gavai *et al.* [18]	73.4	-	-	-	-	-	-	-	0.77	• Use supervised and unsupervised machine learning techniques • Offer a Dashboard • Real-world dataser	• Law Accuracy • Needs more evaluation metrics

2.3 Conclusion

In this chapter, we presented the background of our research topic and its related work. In the background part, we defined the essential concepts used in this thesis, with an emphasis on user profiling, cyber-persona identification, insider threats, and insider threat detection

challenges. We have then presented text representation techniques that we have used in our data processing and the machine learning and deep learning techniques we have used in our framework. With respect to the related research, we presented a number of relevant works on the user profiling and cyber-persona identification, and insider threat detection. Finally, we have presented insider threat detection techniques. In the next chapter, we will present our insider threat detection framework.

Chapter 3

Insider Threat Detection

In this chapter, we will present our network traffic-based user profiling, cyber-persona identification and anomaly detection framework. First, we provide an overview of our approach in Section 3.1. Second, we detail the feature extraction procedure and dataset preparation in Section 3.2. Finally, we present our methodology in Section 3.3.

3.1 Approach

Our main goal is to detect insider threats from network traffic. We model the normal behaviours of an organization's current insiders and detect possible deviations from the profiled behaviour. The detection is completely passive, without installing any software agent in the insiders' devices. To this end, we perform user profiling and cyber-persona identification for the active users in an organization.

The overall architecture of our solution is illustrated in Fig. 3. As can be seen, the raw data received from our partners (details in Section 4.1.1) is fed to the *Network Traffic pre-processing and Labeling* module to pre-process the data, and further label it (details in Section 3.3.1). We assign a tuple of labels consisting of ⟨0,user_id,cyber_persona_type⟩ to each network flow generated by an insider.

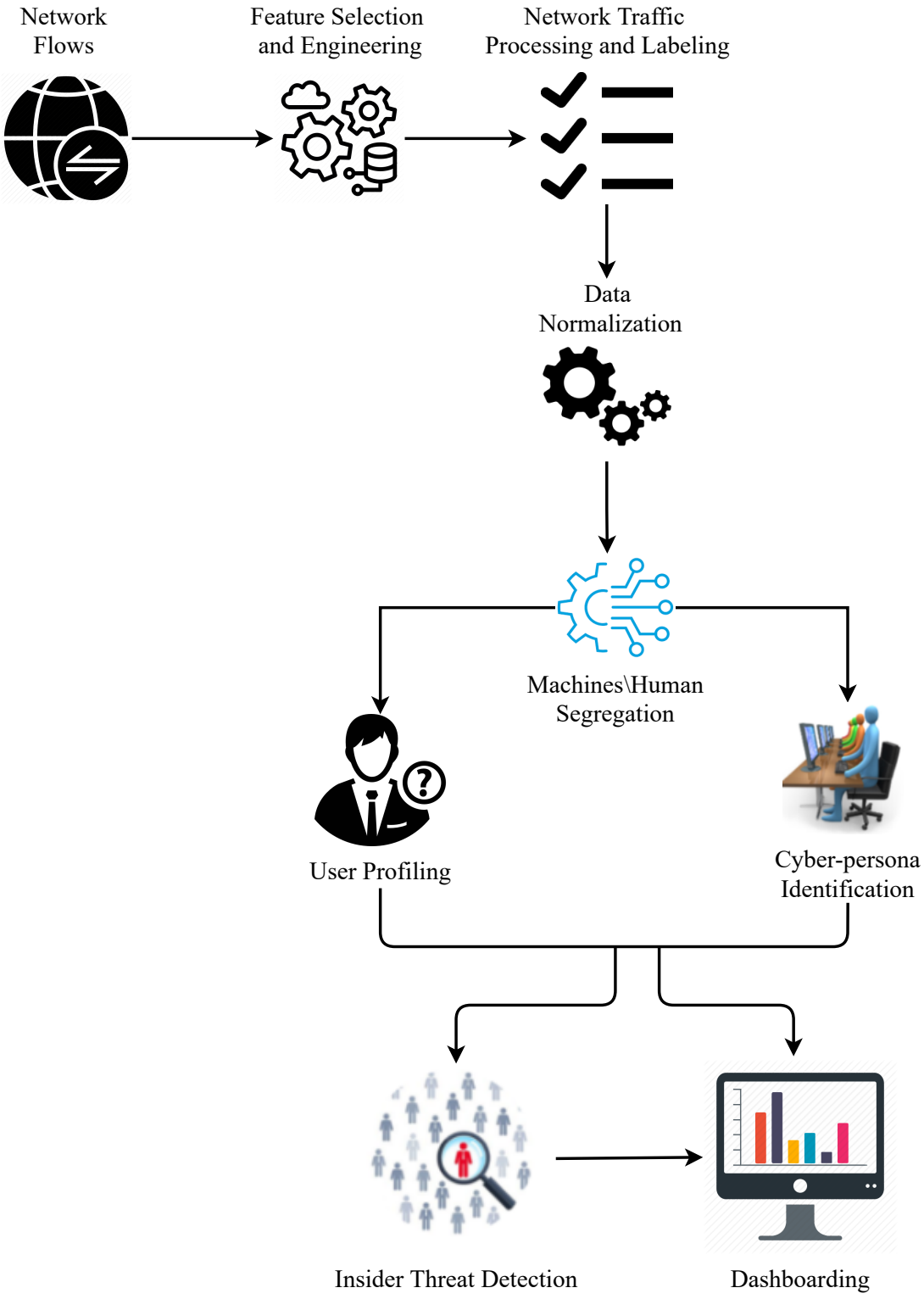


Figure 3: Approach Overview

However, if network flows are generated by machines (e.g., routers, etc.), only a single label will be assigned to the flow ($\langle$`0,NaN,NaN`$\rangle$).

Next, in the case of single flow-based detection, *Features Selection and Engineering* module extracts additional flow features, such as the flow `duration` and normalizes the data. In other experiments, we group the above flows in windows of flows, where each such window contains from 1 to 50 flows. Moreover, for each feature within the window, we compute its mean and standard deviation, and we concatenate them with the representation of the hostnames within the window.

The full process, as well as the data normalization, are described in Section 3.2. Since our primary goal is to model users' network behaviour, only the network flows generated by human users are considered. Therefore, we design a *Machine/Human Segregation* module to automatically segregate flow data generated by machines from the flow data generated by users. After the segregation, we feed user-related data in parallel into the *User Profiling* module, responsible for the identification of the insider's `user_id` (Section 3.3.3 part (i)), and into the *Cyber-Persona Identification* module, responsible for the detection of the user's cyber-persona type (Section 3.3.3 part (ii)). In both of the modules mentioned above, different machine learning models are employed, and their results are compared to highlight the most suitable for our problem. Moreover, single flows, as well as windows of flows, are considered with and without extracting embeddings from the hostnames visited by the insider.

Finally, in the *Insider Threat Detection* module, we flag anomalies by analyzing user behavioural deviations, which include the deviation from the user-specific profile or deviation from the cyber-persona type.

3.2 Features

In this section, we provide details about our features, normalization and data re-sampling.

Table 3: Flow Feature Description

Feature	Description	Feature	Description
id	A unique ID for identifying a flow	local_port	Local port
other_port	Destination port	local_mac	MAC address of local device
last_seen_at	Timestamp for the last time the flow was seen	local_ip	IP address of the local (LAN) device
other_ip	IP address of remote system	other_ip_geo	Geolocation of destination IP
group_name	Cyber-persona type	protocol_name	Protocol detected by*Netify*[1]
protocol_detection _method	Detected protocol using deep packet inspection	application_name	Application generating traffic
application_detection _source	Application categorization based on host-names and address blocks	first_seen_at	Timestamp for the first time the flow was seen
total_local_bytes	Total number of local bytes typically "upload" bytes	total_other_bytes	Total number of other bytes
total_bytes	Total number of bytes	total_packets	Total number of packets
hostname	Visited Hostname		

3.2.1 Feature Selection

The network traffic data provided by our partners contains 19 features. An overview of the raw data features and their description is provided in Table 3. Before employing these features, some pre-processing (discarding some of the features) is needed. For instance, `local_ip` and `local_mac` are discarded since they can also be used as labels. Finally, the features with no change in the data (identical values) such as `id` are discarded as well. Raw data where the values are missing are similarly discarded. With respect to feature extraction, a new feature (`duration`) is calculated from the `first_seen_at` and `last_seen_at`. Moreover, the *mac-vendor-lookup* API[2] is employed in order to extract `Device_type` and `MAC_vendor` from a given MAC address. Finally, the

[2] `https://pypi.org/project/mac-vendor-lookup/`. accessed on November 15, 2020

`Application_category` and `Application_description` are extracted from `hostname` using a mapping provided by our partners. At the end, we employ 21 features, as shown in Table 4. The labels (`group_name` and `user_id`) are provided in a mapping dictionary, as discussed in Section 3.3.1. In the case of `is_machine`, used as a label in the machine/human segregation module, its value is set to zero if the traffic flow has the above two labels (traffic generated by a user) and one otherwise (machines generated traffic).

Table 4: Used Features after Pre-processing

Selected Features	Selected Features
other_ip	other_ip_geo
protocol_name	protocol_detection_method
application_name	MAC_vendor
hostname	first_seen_at
application_detection_source	last_seen_at
mac_vendor	local_port
other_port	duration
total_local_bytes	total_other_bytes
total_bytes	total_packets
application_category	application_description
device_type	

3.2.2 Data Normalization

Initially, we generate a mapping of nominal/categorical feature data to integer values. In this pursuit, we construct a feature dictionary mapping between each categorical value and a corresponding integer value. Then, we employ a Min-Max scaling technique to normalize integer values into $[0, 1]$ interval, using the following formula:

$$x' = \frac{x + min}{max - min} \tag{1}$$

Where x represents a number in a given data record element while min and max represent the minimum and maximum numbers across all corresponding record elements in the data.

3.2.3 Handling Imbalanced Data

From our analysis, we notice that some profiles/cyber-persona types have a small number of flows, not enough for appropriate use in machine learning. Thus, the corresponding data needs to be discarded. To solve this issue, we have used *user_id* and *cyber_persona* types, generating a reasonable number of flows, enough to properly apply machine learning. With respect to the data used in the cyber-persona identification module, we select the ones generating more than $120,000$ network flows. These represent 30 devices and 17 persona types. With respect to user profiling, we retain the profiles generating more than $120,000$ flows, representing 16 devices and 16 profiles. After selecting data for both cyber-persona identification and user profiling, we notice that the data is not balanced, affecting our trained models' performance. According to the literature, up-sampling and down-sampling techniques [24] are recommended. The *Up-sampling* technique involves data duplication, applied on the class data where the number of instances is comparatively small. In contrast, the *Down-sampling* technique involves data deletion, applied on the class data where the number of instances is relatively high.

Up-Sampling. To address the data imbalance issue, we conduct an empirical analysis and comparison between the two techniques. According to the obtained results, the up-sampling technique outperforms the down-sampling one. Therefore, we up-sample our data using SMOTE python library[3]. Figure 4 presents the histogram of flows per cyber-persona types on the small dataset after and before up-sampling.

[3] `https://imbalanced-learn.readthedocs.io`, accessed on January 28, 2021.

3.2.4 Flow Windowing and Embeddings Extraction

In the extension of our approach, we consider the aggregation of sets of network flows generated by the same user for both cases: user profiling and cyber-persona identification. The number of flows per window is empirically determined based on a grid search that we conducted during our experiments. We also extract the context of the hostnames visited by the insiders using *fasttext*[4], a skip-gram model-based method for generating word representations [8]. For each flow window, we first start by computing the mean and the standard deviation of each one of the flow features, which are then concatenated with the embedding generated from the hostnames belonging to the same flow window using *fasttext*. The resulting feature vector will be fed to each one of our models (decision trees, random forests, FFNN, CNN, LSTM and ensemble) in each module.

3.3 Methodology

In this section, we present the details of our approach. To apply supervised machine learning techniques, we label the received network flows designated for model training. However, once our framework is in production, the labelling step is skipped, and the processed flows are fed directly to the Machine/human segregation module, which outputs whether a device used by an employee generates the flow. If so, the flow data will be fed in parallel to the user profiling and cyber-persona identification modules to respectively detect the user id and cyber-persona type (role/position in the company). The results of the aforementioned three steps represent a tuple, as follows: (`is_machine`, `user_id`, `cyber_persona_type`). After identifying the insider's cyber-persona type, the corresponding network flows are fed to the anomaly detection module to detect abnormal behaviour. The main steps of our framework are presented in Algorithm 1.

[4] `https://fasttext.cc/`, accessed on January 28, 2021.

Algorithm 1: Insider Threat Detection

```
input : network_flows_queue
while network_flows_queue not empty do
    flow ← next(network_flows_queue)
    // Machine/human segregation model
    is_machine ← segregate.predict(flow);
    if not(is_machine) then
        // User profiling model
        user_id ← get_profile.predict(flow);
        // Cyber-persona identification model
        cyber_persona_type ← get_persona.predict(flow);
        // List of outlier detection models for each
           cyber-persona-type
        persona_alert ← outlier[cyber_persona_type].predict(flow);
        // List of outlier detection models for each User
           Profile
        user_alert ← outlier[user_id].predict(flow);
        if persona_alert or user_alert then
            alert ← Anomalous;
        else
            alert ← Normal;
        // Save the network flow into elasticsearch
        Save (flow, user_id, cyber_persona_type, alert)
```

3.3.1 Network Traffic Pre-processing and Labeling

Since we aim at solving a classification problem, we first start with labelling our datasets. To this aim, our partners provide us with two dictionaries:

- **Profiling Dictionary.** The dictionary's keys are the `mac_address`' of the devices used by human users, and the values are the `user_ids`'.
- **Cyber-persona Dictionary.** The dictionary's keys are the `mac_address`' of the devices used by human users and the values are the `group_name`' (cyber_persona_types).

We utilize the above information and assign labels to the network traffic flows as follows:

- *Traffic type* (`is_machine`): A binary label value, which is 0 if the flow originates from a device used by an insider, and 1 if the flow originates from a machine.

- *Profiling Label* (`user_id`): A unique positive integer label value representing the *ID* of the user using the device that the flows are originating from.

- *group_name* (`cyber_persona_type`): A unique integer label value representing the persona type of the insider using the device that the flows are originating from.

Therefore, the data coming from a user is labelled with a tuple ⟨`0,user_id,cyber_persona_type`⟩. The dataset contains some MAC addresses that have no corresponding entries in the provided dictionaries. Therefore the related data flows are considered as machine-specific traffic. We assign them the label ⟨`1,NaN,NaN`⟩, which represents machines.

3.3.2 Segregating Human from Machine Traffic

We aim to automatically segregate network traffic generated by users from the traffic of other machines and IoT devices, such as printers, routers, etc. The main reason for developing this module relates to the fact that our goal is to first profile insiders, identify their cyber-persona types and then flag their behavioural deviations. Thus, machine-related traffic should be discarded not to affect the results of our models. We feed labelled flow data to this module; if the flows are generated by machines, they will be discarded automatically; otherwise, we feed to the next modules (user profiling, persona identification and insider threat detection). This module trains different classical supervised machine learning algorithms (decision trees, random forests) and widely used deep learning models (Feed Forward Neural Network (FFNN), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM)) in order to compare their performances and experimentally identify the best-suited model. We detail the configurations of the different hyper-parameters

of models we use in the experimental setup provided in Section 4.1.

3.3.3 User Profiling and Cyber-Persona Identification

We aim at timely detecting insider threats when an insider's network behaviour deviates from its normal behaviour. To this end, we consider two possibilities:

(i) *User Profiling*: Profiling the users based on their network traffic allows to fingerprint their specific behaviours. This includes visited websites, utilized email services, time and duration of the connections, the volume of the data, etc. To this end, we design and implement a network traffic-based user profiling system to discriminate the traffic generated by each network user. The system feeds the network flows to classification algorithms to predict the `id` of the users who generate the flows. We use the same type of classifiers as the machine/human segregation module.

(ii) *Cyber-Persona Identification*: Identifying the cyber-persona types based on the users' network traffic allows fingerprinting their role-specific behaviours. In this module, we model the behaviours of users based on their roles within an organization (e.g., Developer, C-level, Student). We assume that users belonging to the same cyber-persona type are generally working on similar tasks. Consequently, they should have, in general, a shared network behaviour. For instance, the developers would utilize more network traffic compared to human resources personnel. We use the same type of classifiers as the user profiling module.

Those two modules outputs the *user_id,* and *cyber_persona_type* of the insider generating the flows. We feed three types of inputs to the user profiling and cyber-persona identification modules:

(i) When considering single flows, we feed the machine learning models with the 23 normalized flow features.

(ii) When considering single flows with embeddings, we feed the machine learning models with 123 normalized flow features (the extra 100 features represent embeddings resulting from feeding the *hostnames* to the *fasttext* module).

(iii) When considering windows of flows, we feed the models with the average and the standard deviation of the 23 flow features within the flow window, representing 46 new features. Also, we use the 100 embeddings as previously mentioned in (ii). Thus, we end-up with 146 features.

3.3.4 Insider Threat Detection

In this module, we use data corresponding to different cyber-persona types to flag anomalies. With respect to the small dataset, we use different network flows from an unknown cyber-persona type to test our anomaly detection capability. To this end, we feed the network flows to the FFNN model used in the cyber-persona identification module, which generates two types of outputs: the cyber-persona type and the prediction probability. If the latter is lower than an empirically selected threshold, we flag the network flow as abnormal; otherwise, we consider it normal. The lower the persona prediction probability is, the more suspicious the network activity is.

With respect to the large dataset, we apply unsupervised machine learning techniques to detect any behavioural deviations from the user's normal behaviour (profile) or the behaviour of the group of users he belongs to (cyber-persona type). According to the literature, outlier detection techniques [73] is an excellent choice to solve the above issue. After profiling the user and identifying the corresponding cyber-persona type, we feed the generated flows to this module. The latter isolates the abnormal flows and flags them as suspicious. We use isolation forests [36] to isolate suspicious flows. We use this method since anomalous flows are few and different from normal flows. Therefore, the path length from the tree's root to the leaf containing suspicious flows will be shorter than the normal

ones. We train an isolation forest for each profile and each cyber-persona type. Moreover, We use two other outlier detection techniques: robust covariance and One-class SVM. We present models' parameter and experimental details in the next chapter.

3.4 Conclusion

In this chapter, we have presented in details our user profiling, cyber-persona identification and insider threat detection framework. We first started by explaining our approach. Second, we presented our features engineering, including features selection, data normalization, handling imbalanced data issue and flow windowing and embedding extraction. Finally, we explained in details our methodology. In the next chapter, we will present in details our experiments and the results we have obtained.

Chapter 4

Experimental Evaluation

This chapter explains our experimental setup, followed by the obtained results from our different modules (machine/human segregation, user profiling, cyber-persona identification and insider threat detection). Furthermore, we present a discussion about the results. Finally, we present a screenshot of our dashboard.

4.1 Experimental Setup

All of our experiments are conducted on a server running Debian GNU/Linux release 9.12 with Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz and 64GB RAM. The code is written in python and utilizes *sklearn*[1] and *keras*[2] for data processing and machine learning. The profiling and cyber-persona results are stored in *elasticsearch*[3], and the results are visualized in Kibana dashboards.

[1] `https://scikit-learn.org/stable/`, accessed on January 28, 2021.
[2] `https://keras.io/`, accessed on January 28, 2021.
[3] `https://www.elastic.co/`, accessed on January 28, 2021.

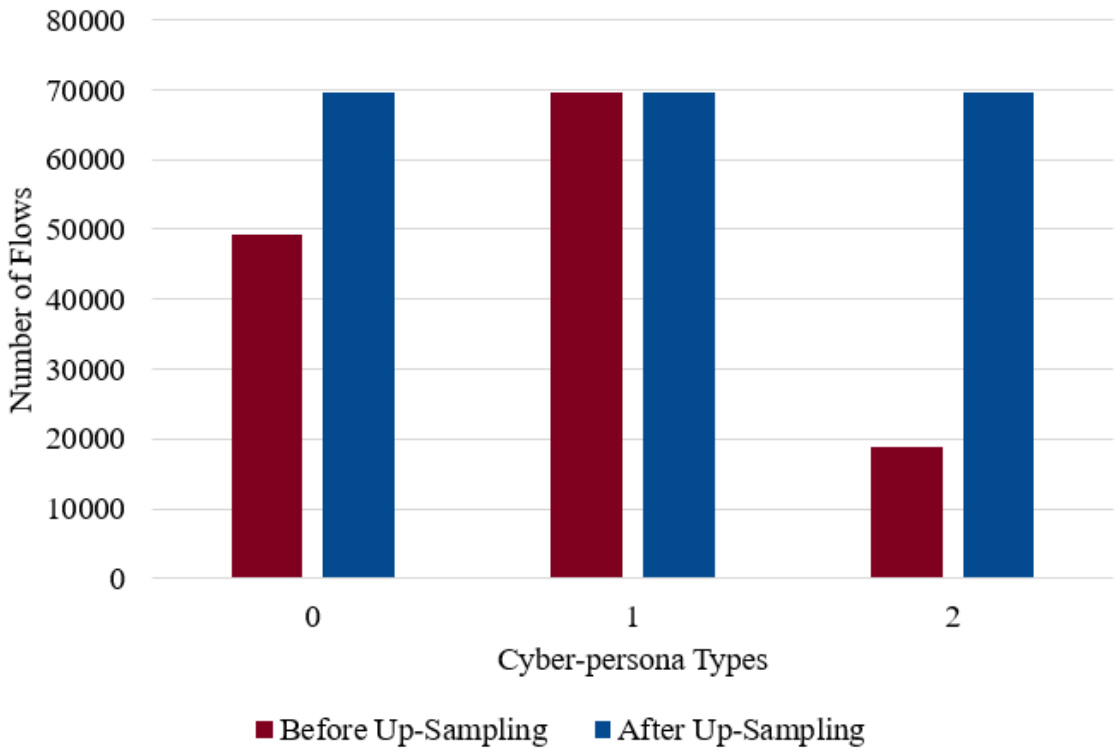
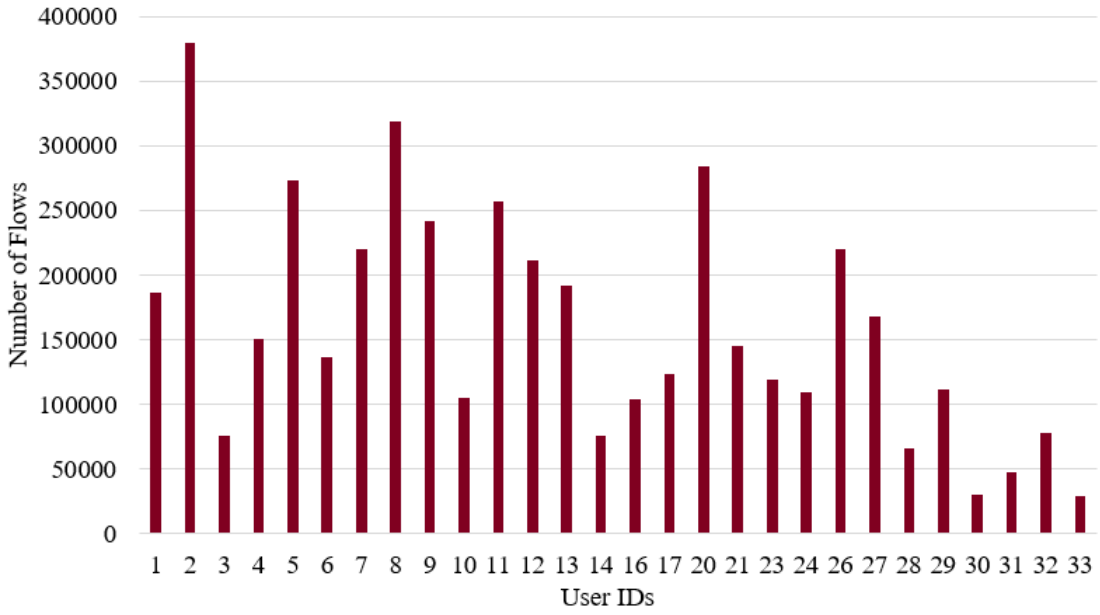


Figure 4: Small Dataset: Flow Distribution per Cyber-Persona Types

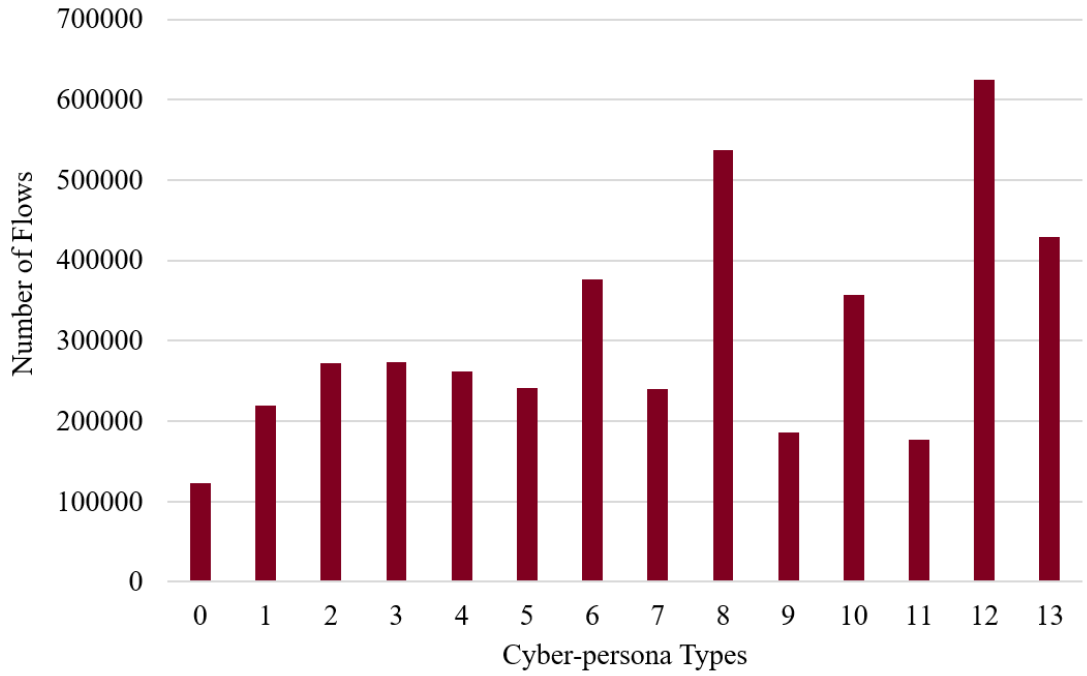
4.1.1 Dataset

Our main goal is to profile insiders, identify their cyber-persona types and flag any behavioural deviation using network traffic. We use real-world network traffic flow data involving a home-generated dataset (named Small Dataset), along with a dataset (called Large Dataset) from the business operations of a company-provided by our industry partners. The corresponding details are presented in the following.

Small Dataset. The small dataset represents traffic collected from a small network and contains approximately 230 thousand flows. This dataset is labelled and separated into four cyber-persona types (three cyber-persona-types will be used for cyber-persona identification while the fourth one is used to test the insider threat detection capability). Moreover, it contains traffic collected from routers, printers or other IoT devices. We use this traffic in our machine/Huma segregation module.



(a) Flow Distribution per User (anonymized)



(b) Flow Distribution per Cyber-persona type

Figure 5: Large Dataset: Flow Distribution

Large Dataset. Our partners collected the large dataset from a real-world business network with a total number of 4,472,391 flows. They also provided us with a dictionary mapping between each user's device's MAC address and the user's *id* and persona type (e.g., Developer, Student, Professor). There are 18 types of persona. Furthermore, the total number of users is 35. However, for confidentiality concerns, the users' *id's* are anonymized and replaced by integer values from 0 to 34. After labelling the received network traffic (Section 3.3.1), we notice that users utilize only 35 devices out of 220 devices connected to the business network. Figures 5a, and 5b, present the number of flows per insider and the number of flows per cyber-persona type, respectively.

4.1.2 Evaluation Metrics

To evaluate the performance of the different components of our framework, we use the accuracy, precision, recall and F_1 Score metrics that are typically used in the literature. These metrics [50] are defined as follows:

$$Accuracy = \frac{TP + TN}{Total\ Number\ of\ Predictions} \tag{2}$$

$$Precision = \frac{TP}{Positive\ Predictions\ Count} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1\ Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{5}$$

Where TP indicates the number of correctly identified users/personas (profiles) as anomalous, TN presents the number of correctly identified profiles as normal, FP indicates the number of incorrectly identified profiles as anomalous, and FN represent the number of anomalous profiles, which have not been identified as anomalies.

4.1.3 Data Splitting

Before training machine learning models, we need to split the dataset. Splitting this last into training, testing, and validation portions is crucial since training the model on a small amount of data will lead to over-fitting instead of a generalization. If the dataset is big enough, multiple data splitting ratios can be adopted. However, if the dataset is limited, data splitting can influence the performance of the models.

In our case, we adopt the splitting ratio mostly used in the literature (80% for training and 20% for testing). Thus, we train and validate our models on 80% of the data, while the remaining data is used for testing. However, we have used 10% of the training data in the case of the Small Dataset and 33% of the training data in the case of the Large Dataset for cross-validation.

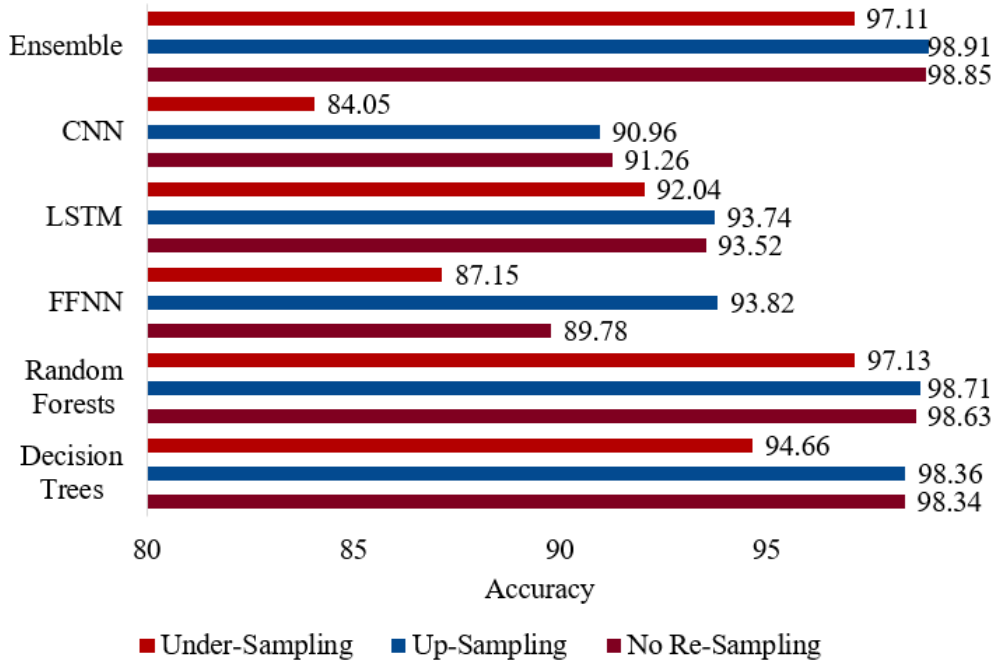
4.1.4 Handling Imbalanced Dataset

In Section 3.2.3, we have discussed techniques used to re-sample imbalanced datasets (mainly up-sampling and under-sampling). Our experimental analysis shows that up-sampling the data improves the performances of our different models, while under-sampling the data decreased the performances of the models. Fig. 6 shows the F_1 score and accuracy obtained from applying up-sampling and under-sampling on the large dataset for both user profiling and cyber-persona identification.

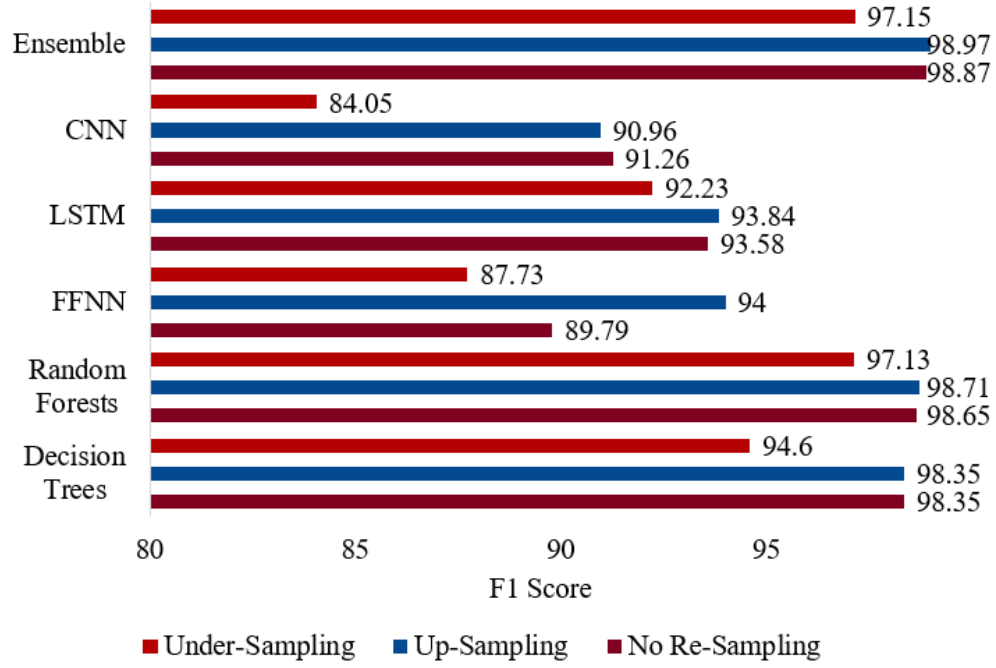
4.1.5 Models and Parameters

With respect to the employed models, we adopt 3 deep neural networks, which are Deep Feed Forward Neural Network[4] (FFNN), Convolutional neural network[4] (CNN) and Long short-term memory[4] (LSTM). The input layer contains a set of neurons equals to the number of features, and the output layer contains a number of neurons equal to the number

[4] `https://keras.io/`, accessed on January 28, 2021



(a) Large Dataset: Acc. for Different Re-sampling Techniques



(b) Large Dataset: F_1 for Different Re-sampling Techniques

Figure 6: Large Dataset: Effects of Data Re-Sampling on Cyber-Persona Identification

Trial 40 Complete [00h 06m 11s]
Val_accuracy: 0.8961639404296875

Best val_accuracy So Far: 0.8968894481658936
Total elapsed time: 04h 05m 16s
INFO:tensorflow:Oracle triggered exit

Figure 7: Small Dataset: Results of Keras Tuner

of desired labels. We use the three deep neural networks in our three modules (machine/human segregation, user profiling and cyber-persona identification). The models are trained during 200 epochs, each with a learning rate of 0.0001. The architectures of the neural networks depend on the problem being solved. We have used Keras Tuner[5] to find out the best architecture of the neural network in our three modules. The result of the application of Keras tuner to find the best FFNN architecture for cyber-persona identification in the small dataset is presented in Fig. 7. It gave us an architecture that employs three hidden layers containing respectively 120, 48 and 16 units.

To compare the results obtained when using deep learning, we also use decision trees and random forests to have a reference base for comparison. Notice that for the two above mentioned models, $random\_state$ is set to 0, and max_depth is set to 100. Finally, we used an ensemble of classifiers to perform better. We have used sklearn ensemble VotingClassifier[6] with the voting parameter set to "hard" to use majority rule voting.

With respect to the anomaly detection part, The following are the outlier detection techniques we have used:

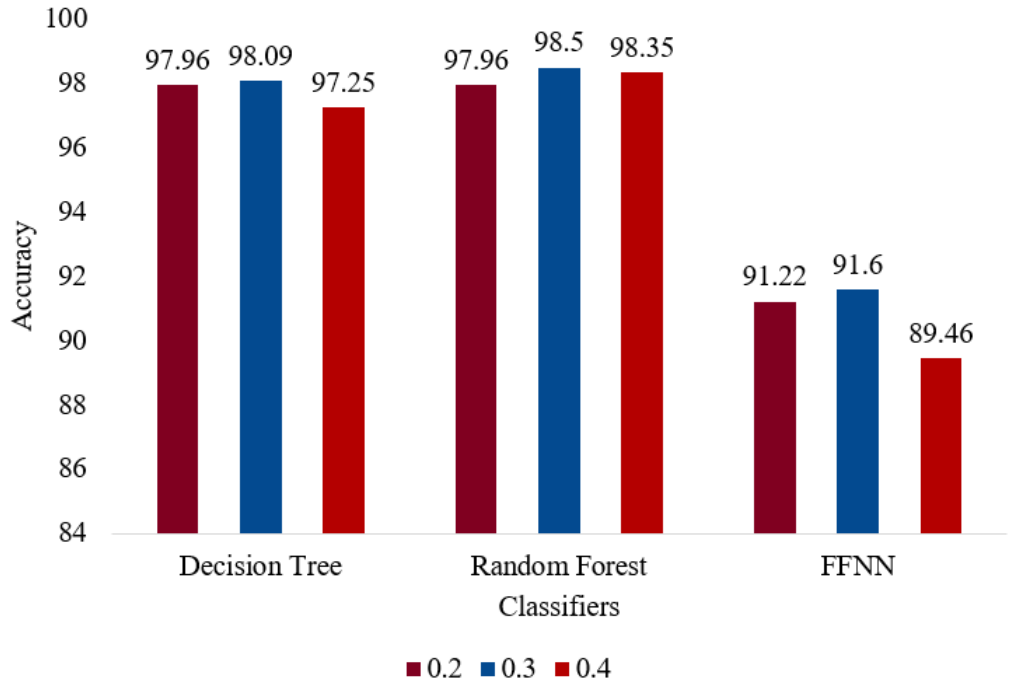
- *Robust Covariance:* with a contamination rate set to 15%.
- *One-Class SVM:* with a ν set to 0.15, a kernel set to *rbf* and γ set to 0.1

[5] `https://www.tensorflow.org/`, accessed on January 27, 2021.
[6] `https://scikit-learn.org`, accessed on January 28, 2021.

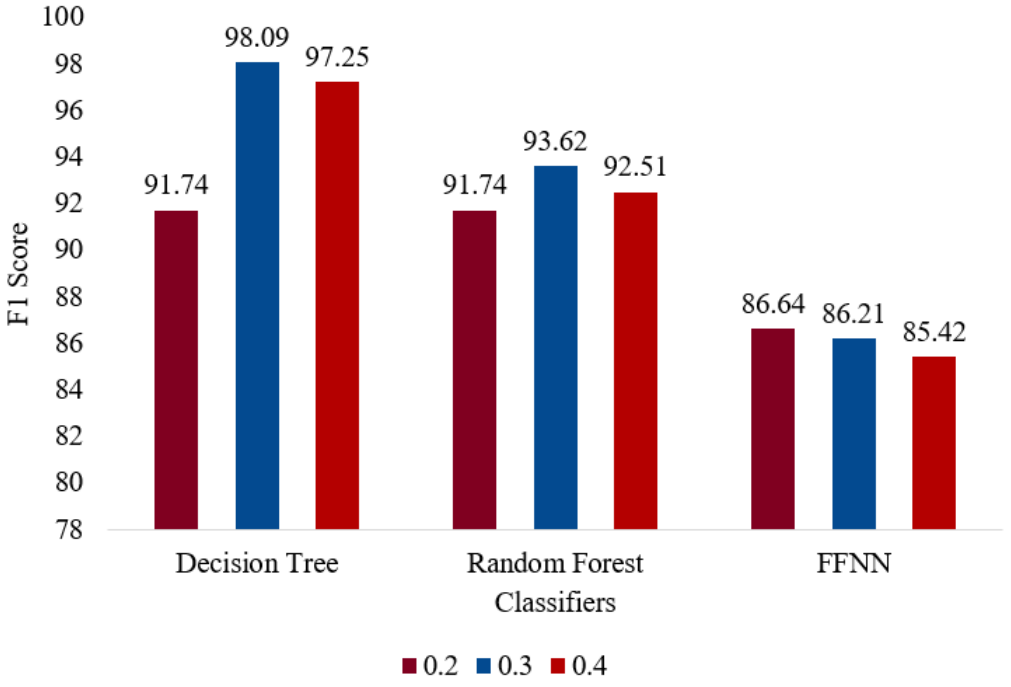
- *Isolation Forest:* with a contamination the rate set to 12%, the random state set to 42, n_estimatiors set to 200 and warm_start set to *true*.

4.1.6 Effects of Data Splitting

To highlight the effects of the data splitting ratios, we conduct extra experiments to measure the three models' accuracy and time complexity. We split the data into different ratios of $80/20\%$, $70/30\%$ and $60/40\%$ for the training and testing. The corresponding accuracy and F_1 score obtained by our three models are presented in Fig. 8. Similarly, the training and prediction time of our three models under different splitting ratios for the training time and the prediction time is illustrated in Fig. 9a and Fig. 9b, respectively.

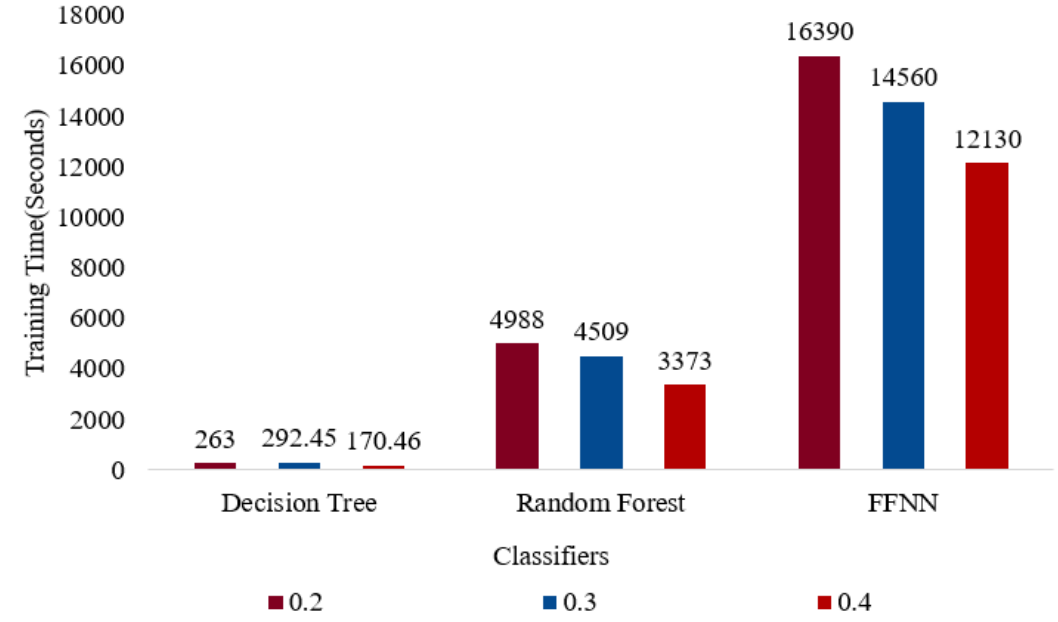


(a) Accuracy per Different Testing Ratios

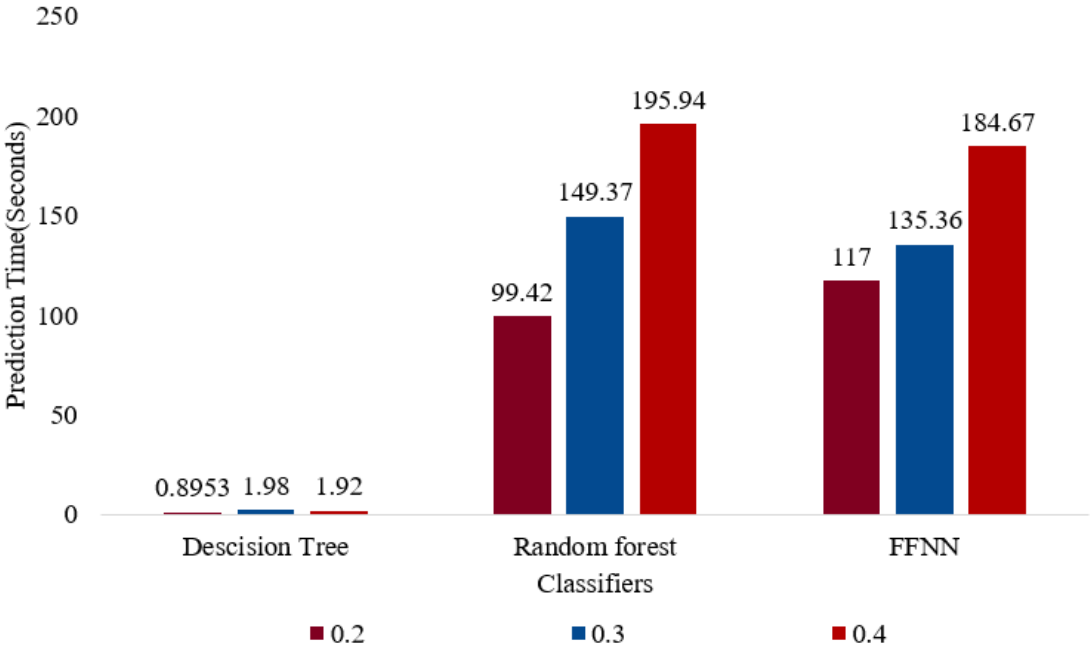


(b) F_1 Score per Different Testing Ratios

Figure 8: Large Dataset: Profiling F_1 Score and Accuracy for Different Testing Ratios



(a) Training Time



(b) Prediction Time

Figure 9: Large Dataset: Models Complexity per Different Splitting Ratios

4.2 Machine/Human Segregation Results

This module aims to answer our first research question to identify the user-specific traffic from the overall device-generated traffic. To this end, we employ three deep learning models (FFNN, CNN and LSTM), decision tree and random forest and an ensemble of models as a binary classifier to segregate humans from machine flow traffic. We present the obtained experimental results on the small and large Datasets in Table 5. Concerning the small dataset, the random forest gave the best results with 97.96% of accuracy and 97.96% of F_1 score. Concerning the large dataset, decision trees gave the best results with 97.95% of accuracy and 97.38% of F_1 score, followed by random forests with 97.90% and 97.32% of F_1 score. Moreover, we notice that having imbalanced data did not negatively affect our machine/human segregation module's performance. Therefore, there is no need to apply up-sampling techniques in this case.

Table 5: Machine/Human Segregation Results

	Small Dataset		Large Dataset	
Models	Acc.	F_1	Acc.	F_1
Decision Trees	97.41%	97.41%	**97.95%**	**97.38%**
Random Forests	**97.96%**	**97.96%**	97.90%	97.32%
FFNN	95.62%	95.62%	95.50%	95.30%
LSTM	96.16%	95.96%	91.94%	91.94%
CNN	95.15%	95.15%	89.97%	89.97%
Ensemble	97.64%	97.64%	97.76%	97.13%

Table 6: Small Dataset: Acc. and F_1 Results using Single Flows

	Before Up-sampling		After Up-sampling	
Models	Acc.	F_1	Acc.	F_1
Decision Trees	75.58%	75.58%	75.49%	75.49%
Random Forests	74.17%	74.17%	73.87%	73.87%
FFNN	73.73%	73.73%	73.67%	73.67%
LSTM	73.66%	66.05%	73.23%	73.23%
CNN	72.71%	72.71%	72.87%	72.87%
Ensemble	**75.59%**	**75.59%**	**75.82%**	**75.82%**

(a) User Profiling

	Before Up-sampling		After Up-sampling	
Models	Acc.	F_1	Acc.	F_1
Decision Trees	86.08%	86.08%	85.26%	77.08%
Random Forests	89.54%	89.54%	87.02%	78.09%
FFNN	**89.94%**	**89.94%**	86.18%	77.6%
LSTM	89.89%	89.89%	88.36%	88.36%
CNN	89.89%	89.89%	**88.85%**	**88.85%**
Ensemble	87.57%	87.57%	85.53%	85.53%

(b) Cyber-persona Identification

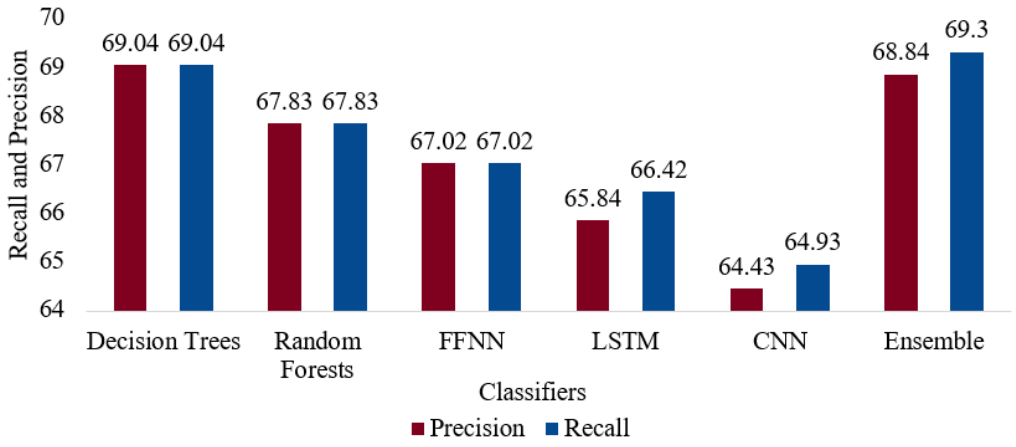


Figure 10: Small Dataset: Profiling Recall and Precision before Up-Sampling

Table 7: Large Dataset: Acc. and F_1 Results using Single Flows

	Before Up-sampling		After Up-sampling	
Models	Acc.	F_1	Acc.	F_1
Decision Trees	97.96%	91.74%	98.52%	98.52%
Random Forests	97.96%	91.74%	94.74%	98.72%
FFNN	93.78%	93.69%	95.69%	95.68%
LSTM	94.8%	89.89%	92.54%	87.7%
CNN	89.05%	89.05%	90.94%	90.94%
Ensemble	**98.86%**	**98.86%**	**98.89%**	**98.89%**

(a) User Profiling

	Before Up-sampling		After Up-sampling	
Models	Acc.	F_1	Acc.	F_1
Decision Trees	98.65%	98.68%	98.36%	98.35
Random Forests	98.35%	98.38%	98.71%	98.71
FFNN	92.95%	92.93%	95.33%	95.32%
LSTM	93.52%	93.58%	93.74%	93.84%
CNN	91.26%	91.26%	90.96%	90.96%
Ensemble	**98.85%**	**98.88%**	**98.91%**	**98.91%**

(b) Cyber-persona Identification

4.3 User Profiling Results

We further answer the second and the first part of our third research question on identifying users based on their network traffic. To this end, we similarly apply the six models on both datasets (home and large datasets) before and after up-sampling the data.

Small Dataset. The obtained accuracy and F_1 score for the small dataset are shown in Table 6. The recall and precision scores before and after up-sampling are shown in Fig. 10 and Fig. 11. As can be seen, the ensemble provides the best results with an accuracy of 75.59% and an F_1 score of 75.59% before up-sampling. Up-sampling the data slightly improves our models' performance, with an accuracy of 75.82% and an F_1 score of 75.82% for the ensemble model. To improve the performance, we extended our experiments by considering windows of flows instead of single flows. We combine the flow windows with the embedding vectors obtained from the application of *fasttext* on the hostnames within

the window. We illustrate the grid search results on the small dataset (in the case of user profiling) in Fig. 12. We considered many window flow sizes before and after up-sampling. Table 10 shows the flow windows sizes providing the best results for decision trees, random forest and neural networks. As can be seen, Feed Forward Neural Network gave the best results before up-sampling, with 77.15% of accuracy and 69.94% of F_1 score when considering 10 flows per window. On the other hand, after up-sampling, Random Forest gave the best results with 93.65% of accuracy and 93.46% of F_1 score when considering 30 flows per window.

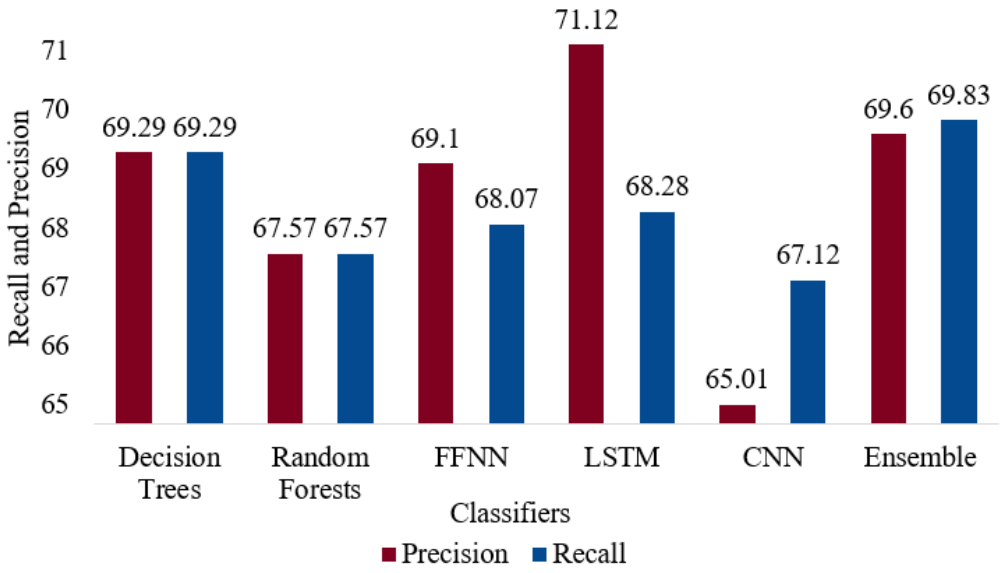
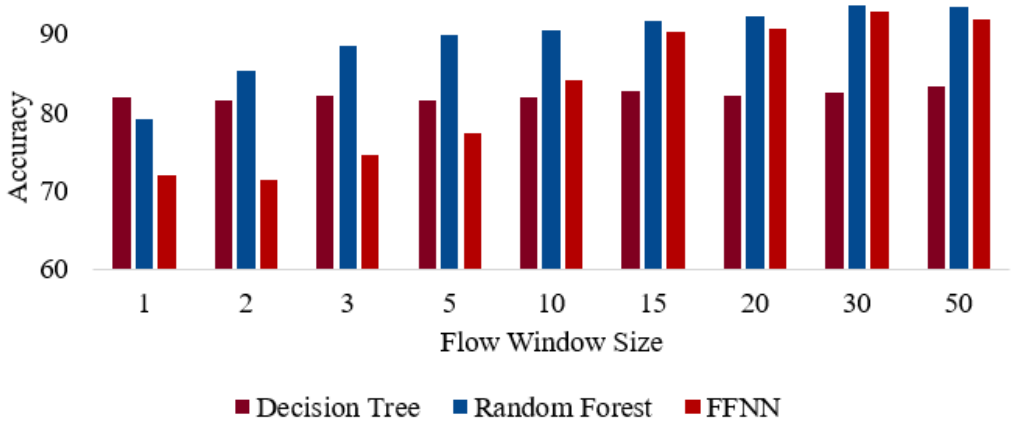
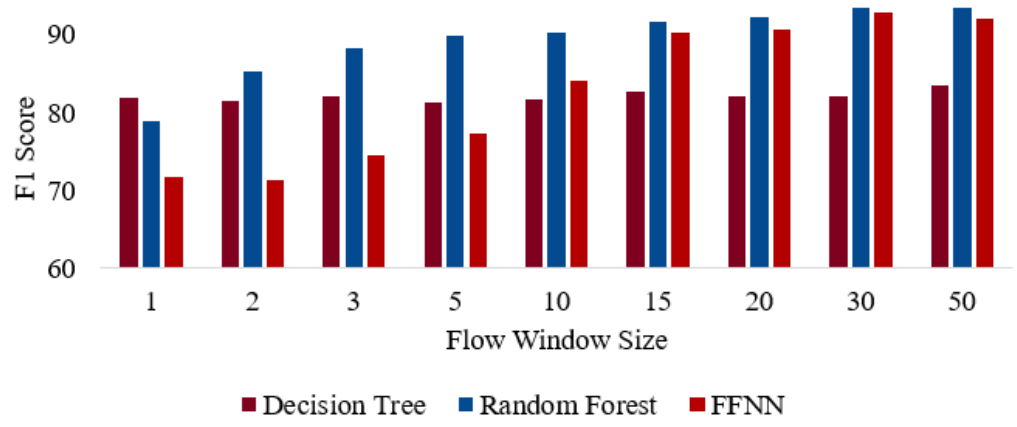


Figure 11: Small Dataset: Profiling Recall and Precision after Up-Sampling



(a) Accuracy per Different Flow Windows Sizes



(b) F_1 Score per Different Flow Windows Sizes

Figure 12: Small Dataset: Profiling Grid Search after Up-Sampling

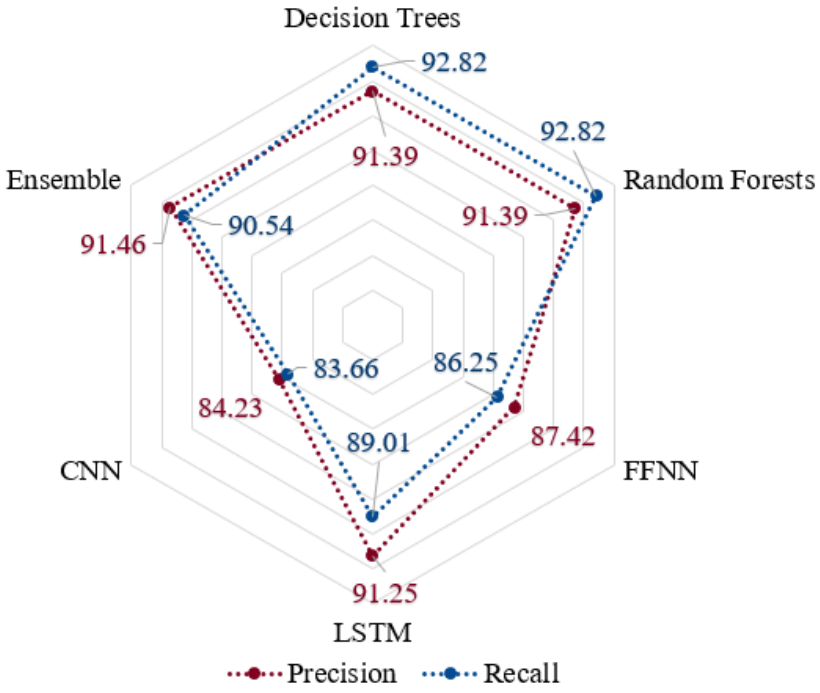


Figure 13: Large Dataset: Profiling Recall and Precision before Up-Sampling

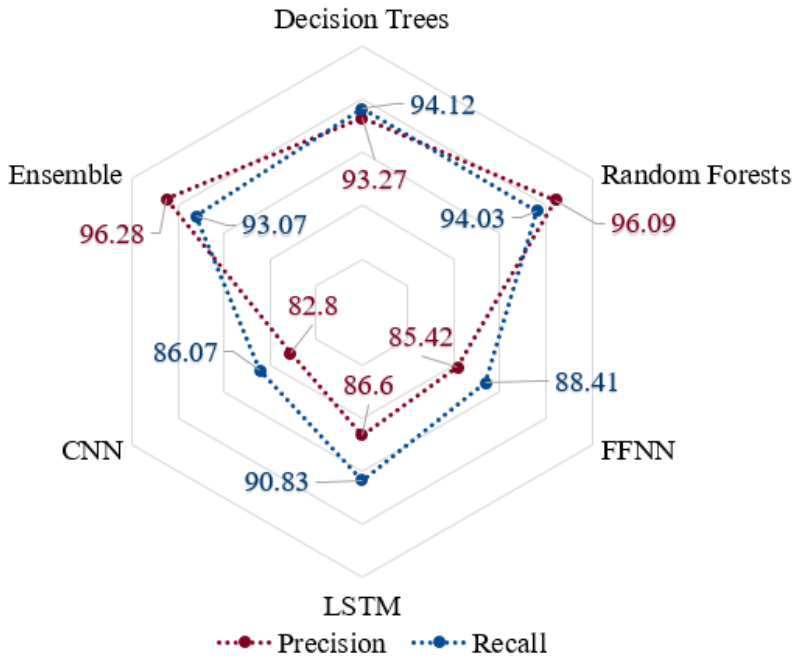
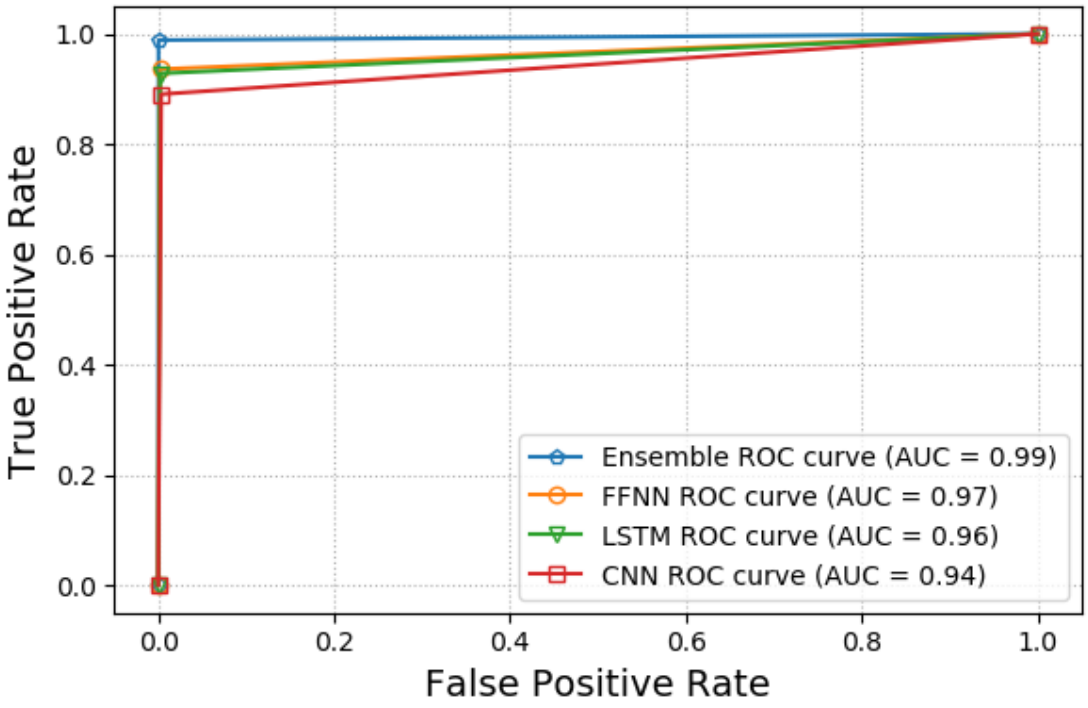
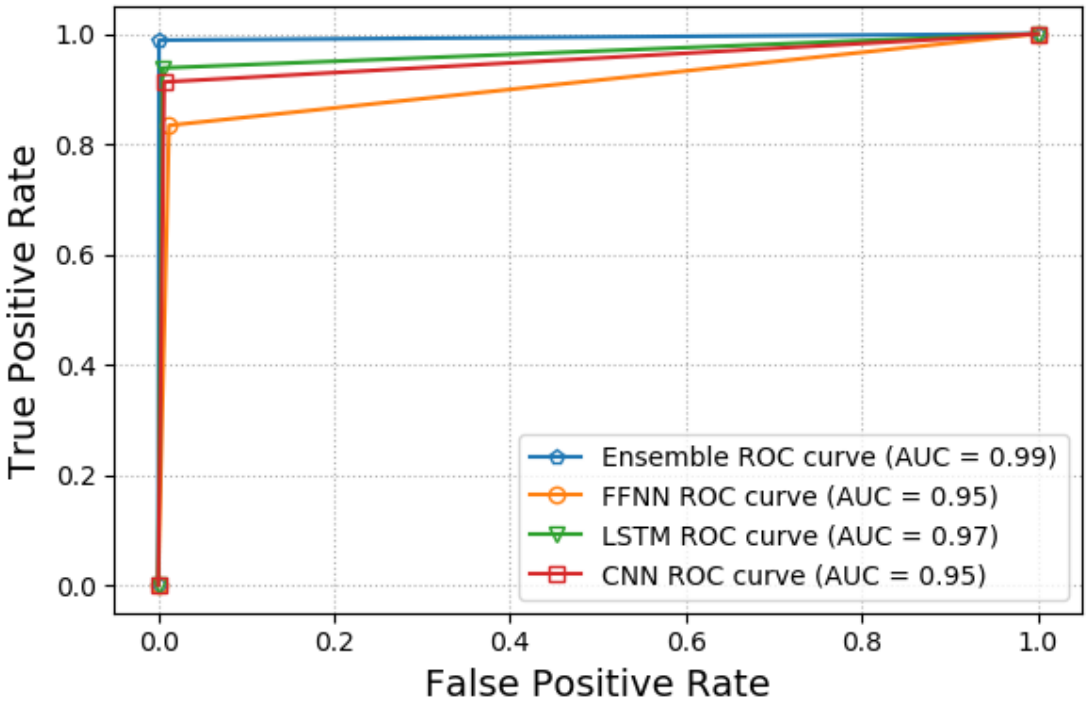


Figure 14: Large Dataset: Profiling Recall and Precision after Up-Sampling

Large Dataset. For the large dataset, Table 7 presents our profiling models' performance results, both before and after up-sampling the data. The precision and recall are shown in Fig. 13 and Fig. 14. The ROC curve is shown in Fig. 15a and the FFNN loss and accuracy for user profiling are illustrated in Fig. 16 and Fig. 17 respectively. The ensemble gave the best results before up-sampling, with 98.86% of accuracy and 98.86% of F_1 score. After up-sampling, the ensemble also gave the best results with 98.89% of accuracy and 98.89% of F_1 score. With respect to the grid search results, Fig. 21 presents the results obtained from applying grid search on the large dataset, considering different flow windows sizes before and after up-sampling. Table 9 shows the flow window sizes providing the best results for decision trees, random forests and FFNN. As can be seen, FFNN gave the best results before up-sampling with 96.07% of accuracy and 94.44% of F_1 score when considering 20 flows per window. After up-sampling, FFNN also gave the best results with 98.56% of accuracy and 98.54% of F_1 score when considering 30 flows per window.

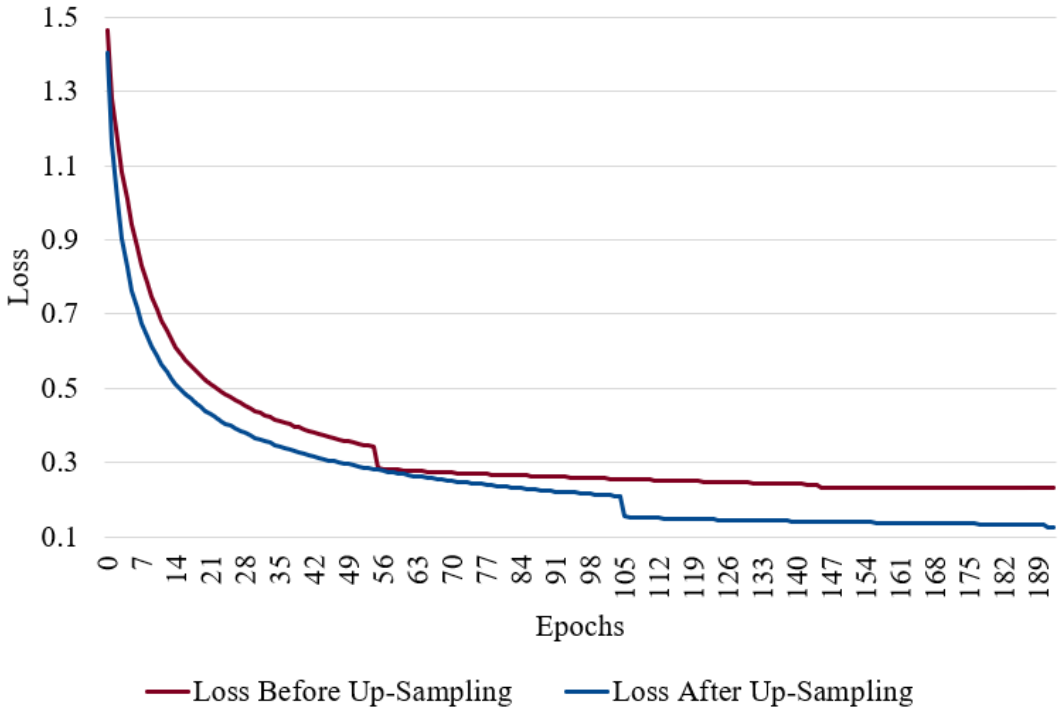


(a) User Profiling ROC Curve

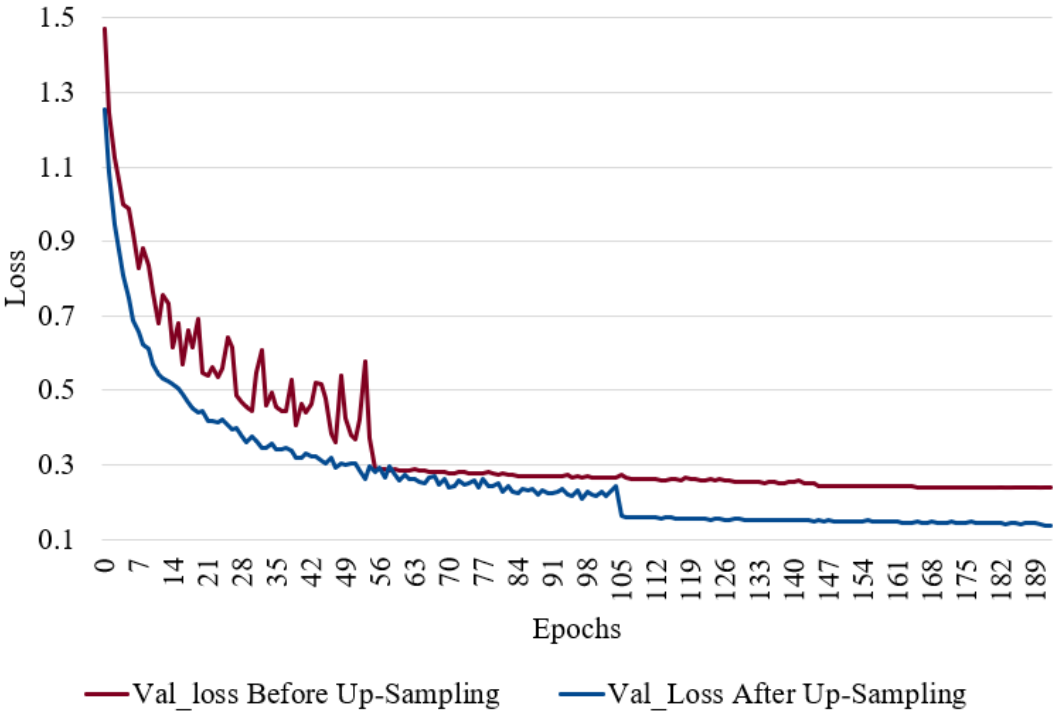


(b) Cyber-persona Identification ROC Curve

Figure 15: Large Dataset: Profiling and Cyber-persona ROC Curve

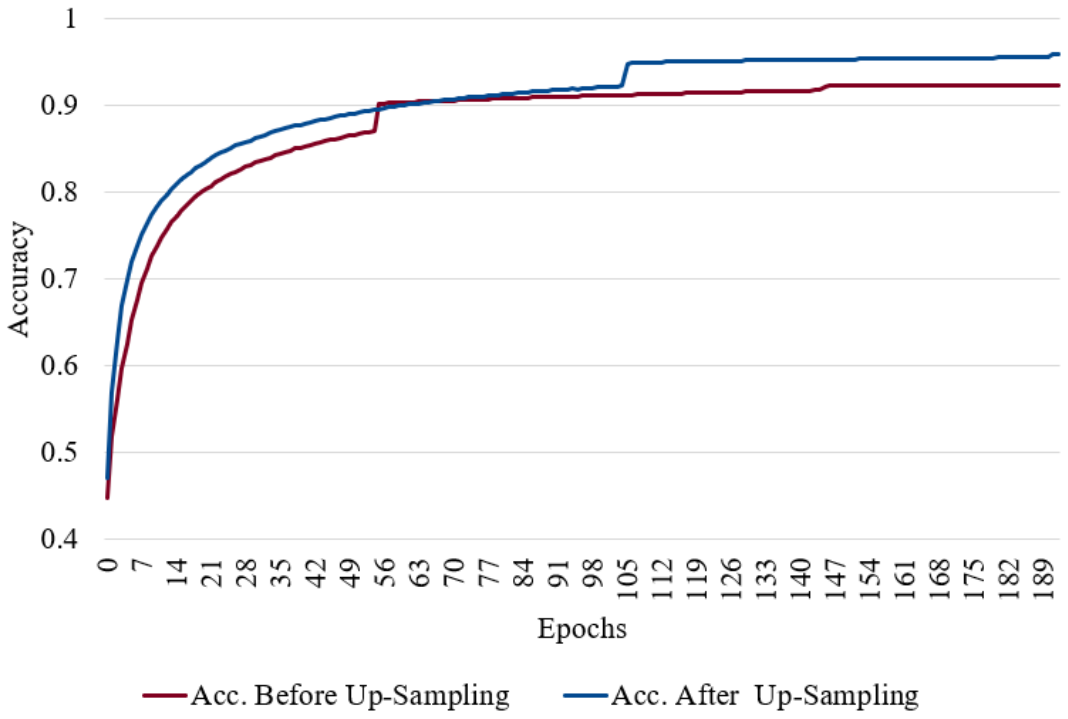


(a) User Profiling model's Training Loss

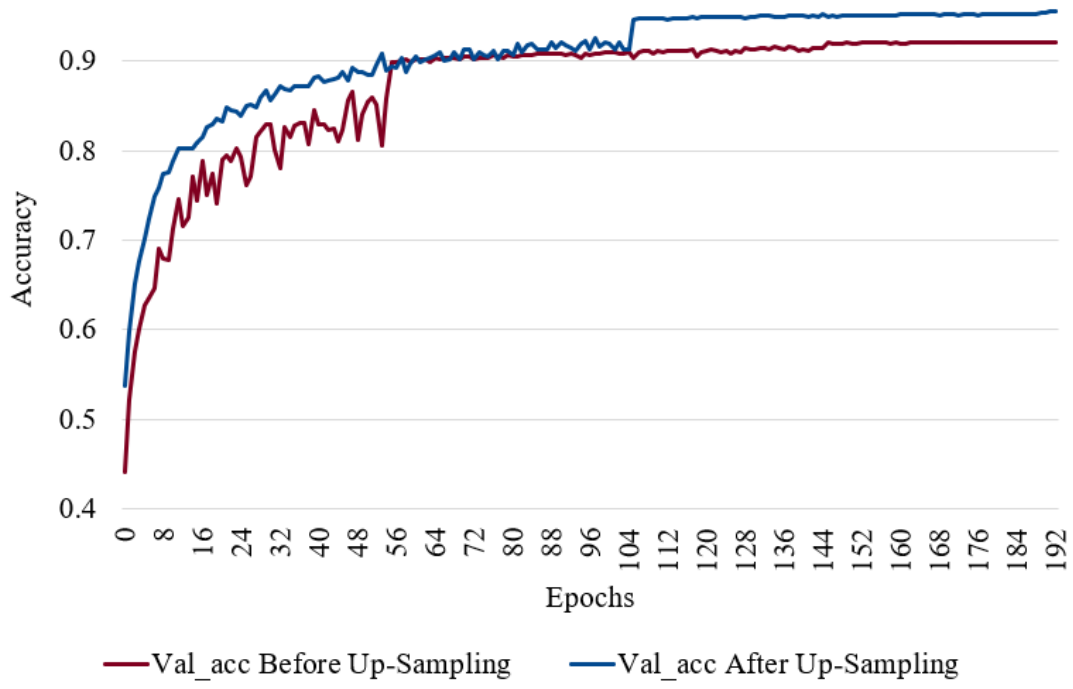


(b) User Profiling model's Validation Loss

Figure 16: Large Dataset: Profiling Training and Validation Loss

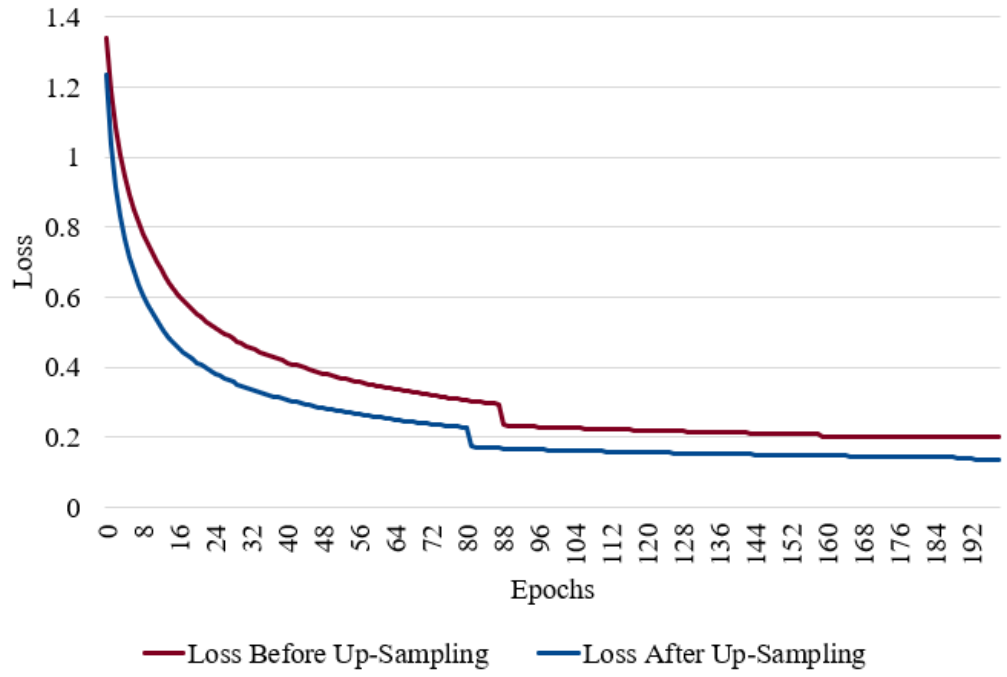


(a) User Profiling model's Training Accuracy

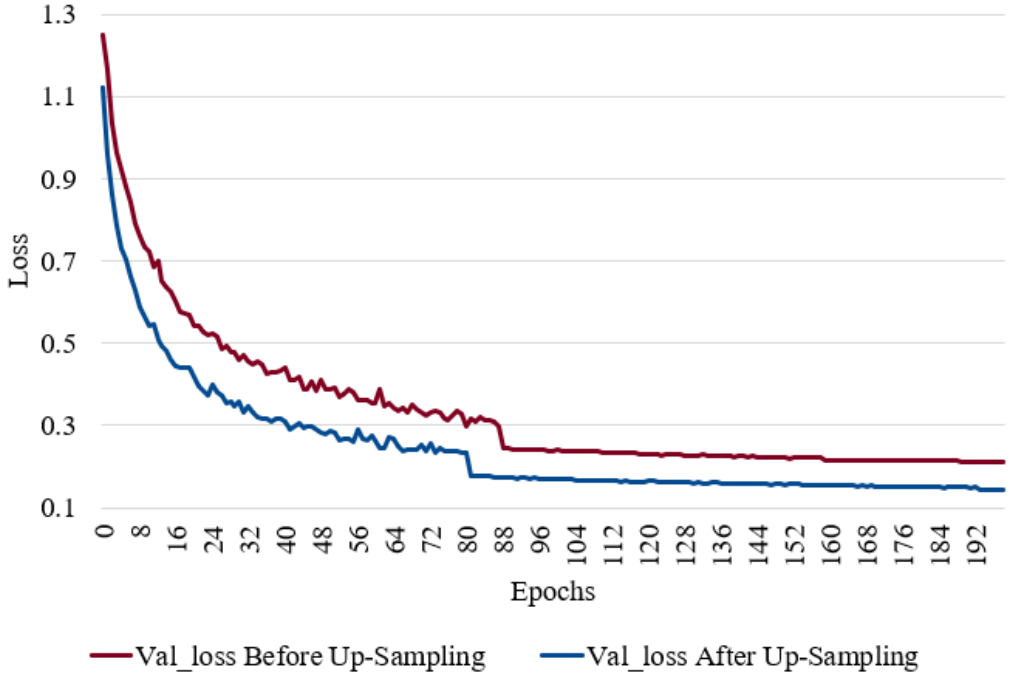


(b) User Profiling model's Validation Accuracy

Figure 17: Large Dataset: Profiling Training and Validation Accuracy

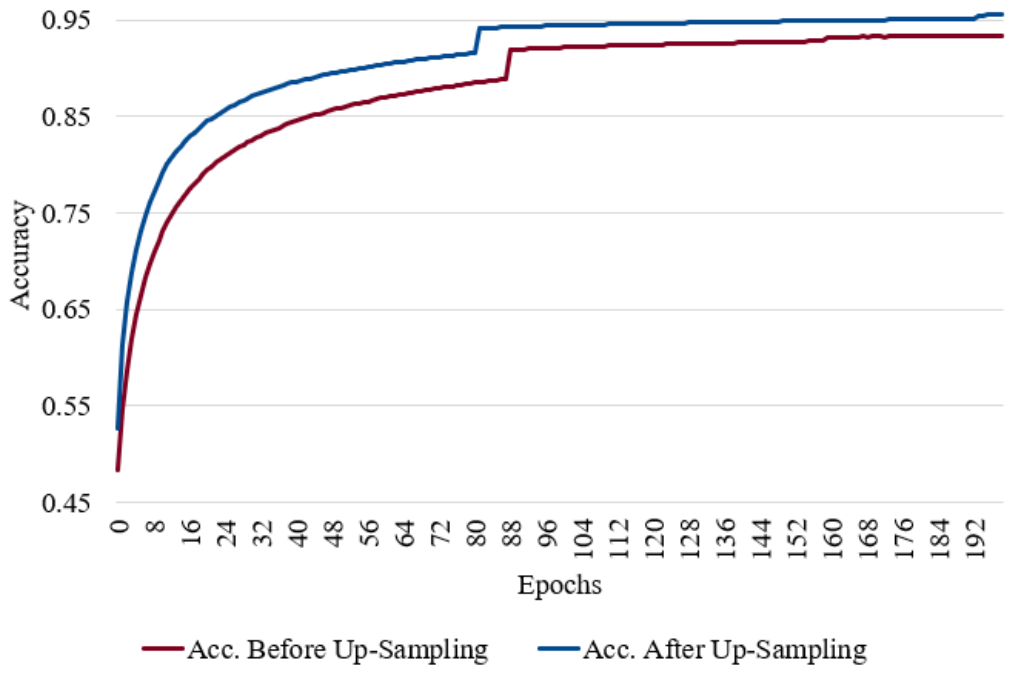


(a) Cyber-Persona Identification Training Loss

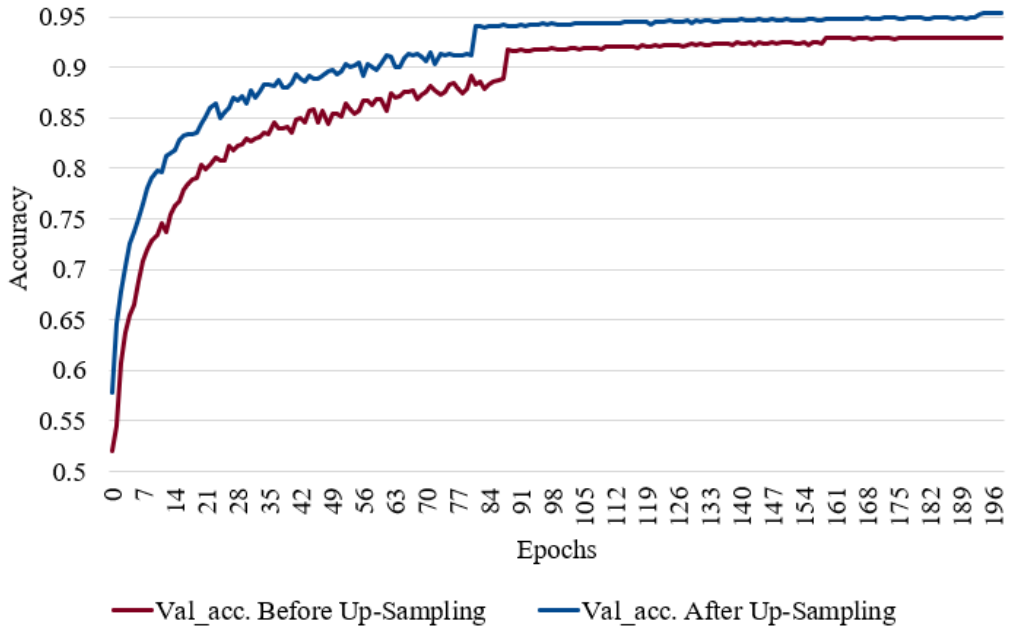


(b) Cyber-Persona Identification Validation Loss

Figure 18: Large Dataset: Cyber-persona Training and Validation Loss

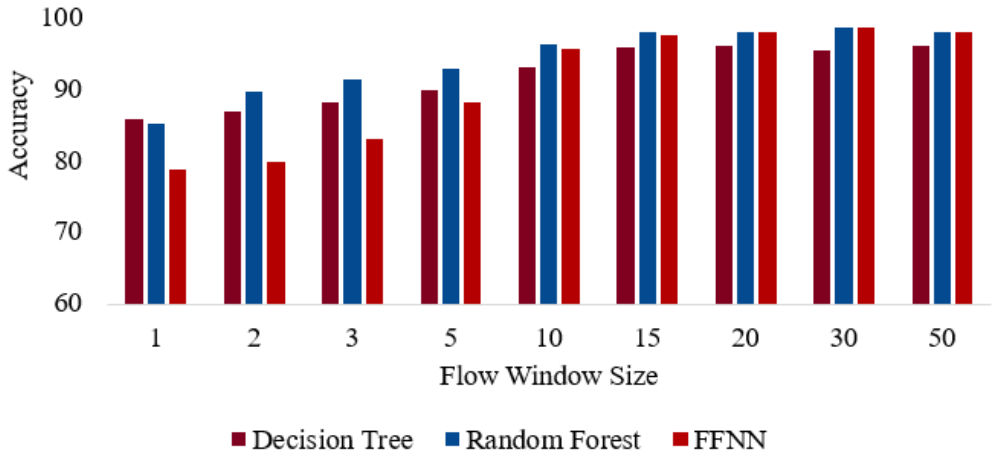


(a) [Cyber-Persona Identification Training Accuracy

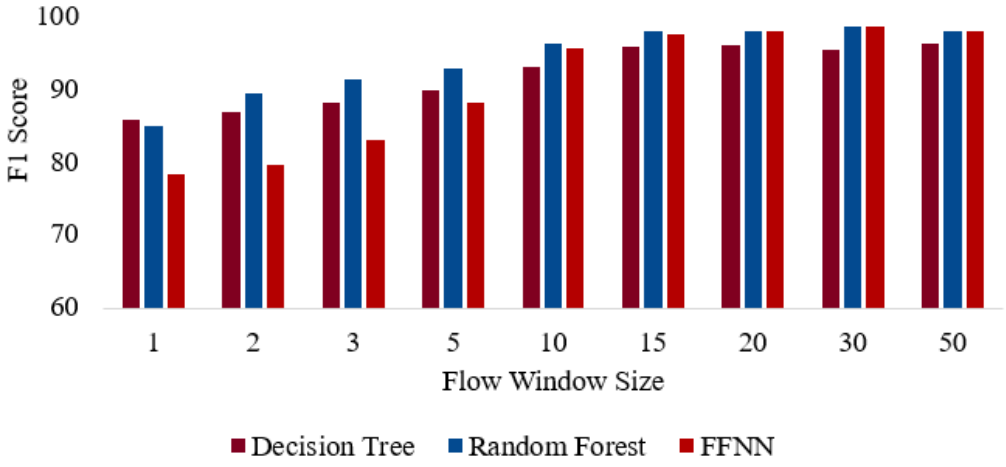


(b) Cyber-Persona Identification Validation Accuracy

Figure 19: Large Dataset: Cyber-persona Training and Validation Accuracy

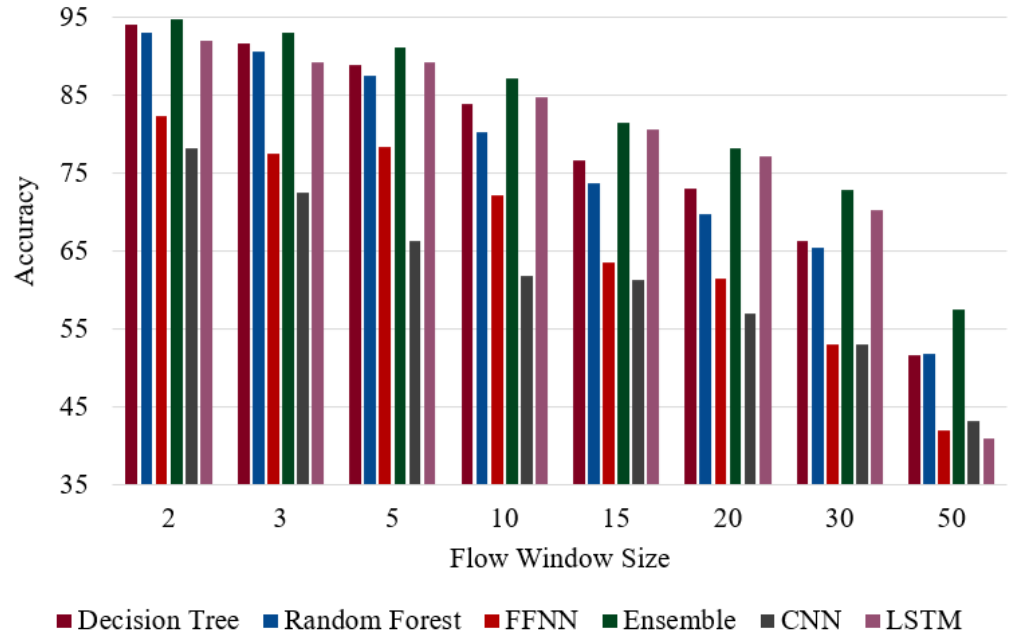


(a) Accuracy per Different Flow Windows Sizes

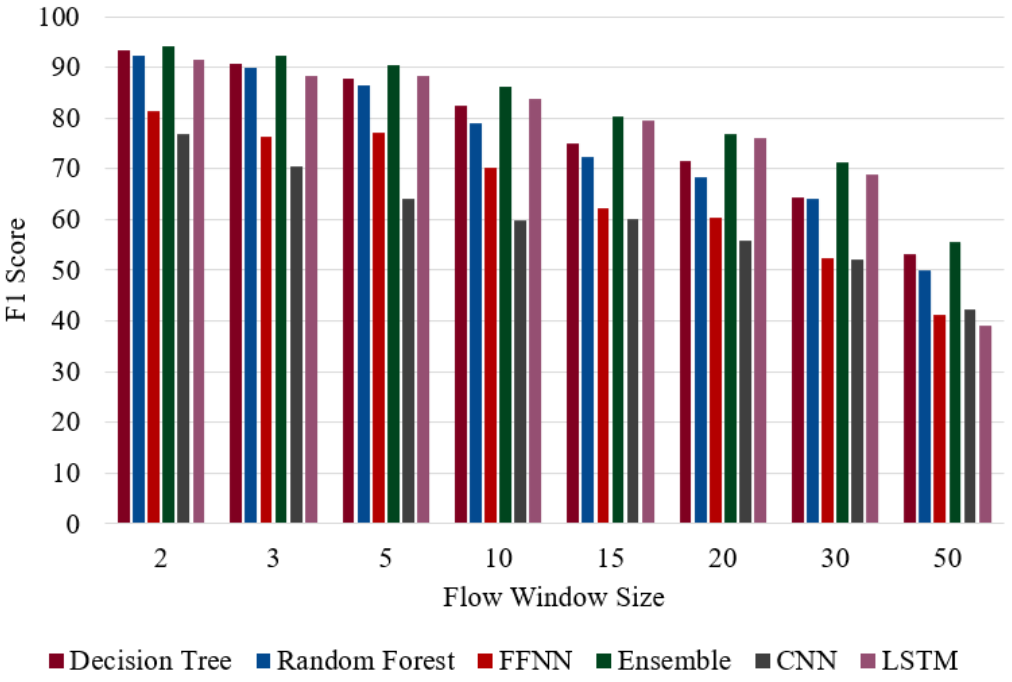


(b) F_1 Score per Different Flow Windows Sizes

Figure 20: Small Dataset: Cyber-persona Grid Search after Up-Sampling



(a) Accuracy per Different Flow Windows Sizes



(b) F_1 Score per Different Flow Windows Sizes

Figure 21: Large Dataset: Profiling Grid Search after Up-Sampling

Table 8: Large Dataset: Acc. and F_1 Results using Single Flows with Embeddings

	Before Up-sampling		After Up-sampling	
Models	F_1	Acc.	F_1	Acc.
Decision Trees	98.37%	98.48%	98.17 %	98.30 %
Random Forests	97.61%	97.79%	97.63%	97.81%
FFNN	86.24%	87.06%	84.43%	85.23%
LSTM	96.94%	97.06%	97.02%	97.14%
CNN	82.00%	83.12%	80.35%	81.56%
Ensemble	**98.59%**	**98.68%**	**98.45%**	**98.55%**

(a) User Profiling

	Before Up-sampling		After Up-sampling	
Models	F_1	Acc.	F_1	Acc.
Decision Trees	98.08%	98.10%	98.20%	98.25%
Random Forests	97.76%	97.79%	97.77%	97.83%
FFNN	87.10%	87.39%	85.33%	85.18%
LSTM	96.04%	96.12%	96.93%	96.98%
CNN	79.06%	79.56%	81.79%	81.44%
Ensemble	**98.35%**	**98.37%**	**98.44%**	**98.47%**

(b) Cyber-persona Identification

Table 9: Large Dataset: Best Results using Flow Windows and Embedings

	Before Up-sampling			After Up-sampling		
Models	Flow Win. Size	F_1	Acc.	Flow Win. Size	F_1	Acc.
Decision Trees	3	93.66%	94.15%	2	93.42%	94.05 %
Random Forests	2	92.61%	93.21 %	2	92.39%	93.1%
FFNN	2	78.96%	80.32%	2	81.3%	82.39%
LSTM	2	92.49%	92.95 %	2	91.49 %	91.78%
CNN	2	75.25%	76.55%	2	76.81%	78.14%
Ensemble	**2**	**95.12%**	**95.49%**	**2**	**94.24 %**	**94.78 %**

(a) User Profiling

	Before Up-sampling			After Up-sampling		
Models	Flow Win. Size	F_1	Acc.	Flow Win. Size	F_1	Acc.
Decision Trees	2	93.72%	93.83 %	2	94.28%	94.64%
Random Forests	2	93.09%	93.2%	2	92.68%	93.03%
FFNN	2	78.86%	79.61%	2	78.58%	78.71 %
LSTM	2	91.49 %	91.78 %	2	90.93%	91.32%
CNN	2	76.43 %	76.96%	3	73.92 %	74.41%
Ensemble	2	**94.85%**	**94.93%**	2	**94.61%**	**94.94%**

(b) Cyber-persona Identification

Table 10: Small Dataset: Best Results using Flow Windows and Embedings

	Before Up-sampling			After Up-sampling		
Models	Flow Win. Size	F_1	Acc.	Flow Win. Size	F_1	Acc.
Decision Trees	10	67.86%	75.76%	50	83.43%	83.23%
Random Forests	10	69.75%	76.77%	30	**93.46%**	**93.65%**
FFNN	10	**69.94%**	**77.15%**	30	92.78%	92.93%

(a) User Profiling

	Before Up-sampling			After Up-sampling		
Models	Flow Win. Size	F_1	Acc.	Flow Win. Size	F_1	Acc.
Decision Trees	20	94.72%	95.78%	50	96.08%	96.04%
Random Forests	50	**95.46%**	**96.36%**	30	98.49%	98.48%
FFNN	20	94.44%	96.07%	30	**98.54%**	**98.56%**

(b) Cyber-persona Identification

Table 11: Small Dataset: Acc. and F_1 Results using Single Flows and Embeddings

	Before Up-sampling		After Up-sampling	
Models	F_1	Acc.	F_1	Acc.
Decision Trees	**69.67%**	**76.08%**	**81.88%**	**81.87%**
Random Forests	65.84%	73.42%	78.91%	79.13%
FFNN	66.55%	74.15%	71.79%	71.87%

(a) User Profiling

	Before Up-sampling		After Up-sampling	
Models	F_1	Acc.	F_1	Acc.
Decision Trees	77.58%	86.24%	**85.65%**	**85.66%**
Random Forests	78.37%	88.25%	84.83%	84.95%
FFNN	**78.34%**	**89.84%**	78.13%	78.63%

(b) Cyber-persona Identification

4.4 Cyber-Persona Identification Results

In this section, we aim to answer the second part of our third research question. Thus, we present the cyber-persona identification results obtained on the small and large datasets.

Small Dataset. We aim to identify three cyber-persona types on the small dataset (see Section 4.1.1 for its description). Table 6 shows the accuracy and F_1 score results, while Fig. 22 and Fig. 23 illustrate the recall and precision scores. As can be seen, FFNN model provide better results with 89.94% of accuracy and F_1 score before up-sampling, while CNN model provide the best results after up-sampling the data with 88.85% of accuracy and F_1 score. Moreover, Fig. 20 present the results obtained during the grid search conducted on the small dataset by considering different flow window sizes. Table 10 shows the flow window sizes providing the best results for decision trees, random forest and FFNN. As can be seen, before up-sampling the data, random forest gave the best performance with 96.36% of accuracy and 95.46% of F_1 score when considering 50 flow per window. However, after up-sampling the data, FFNN gave the best results with 98.56% of accuracy and 98.54% of F_1 score when considering 30 flows per window.

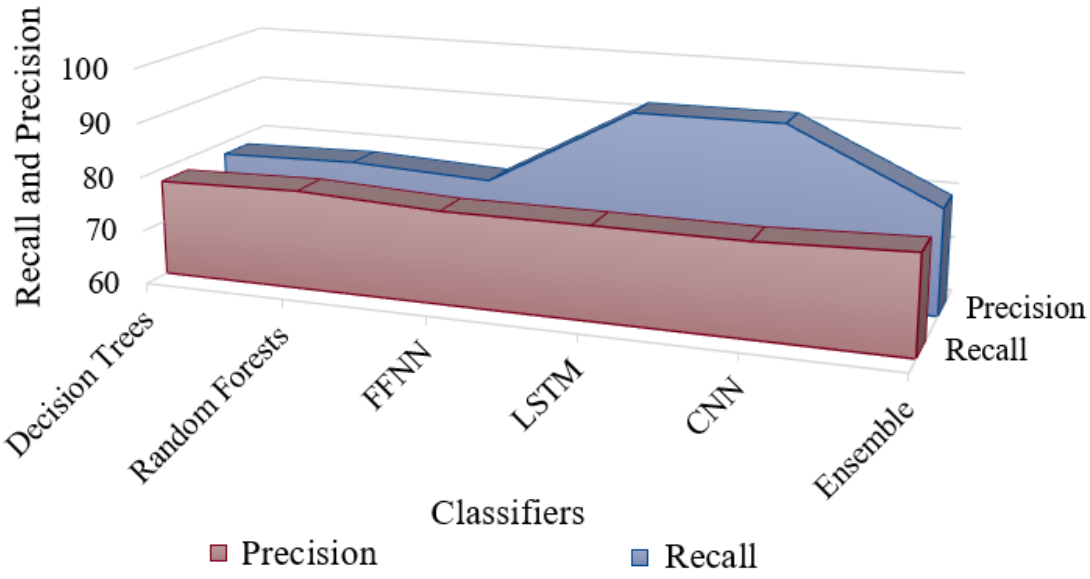


Figure 22: Small Dataset: Cyber-persona Recall and Precision before Up-Sampling

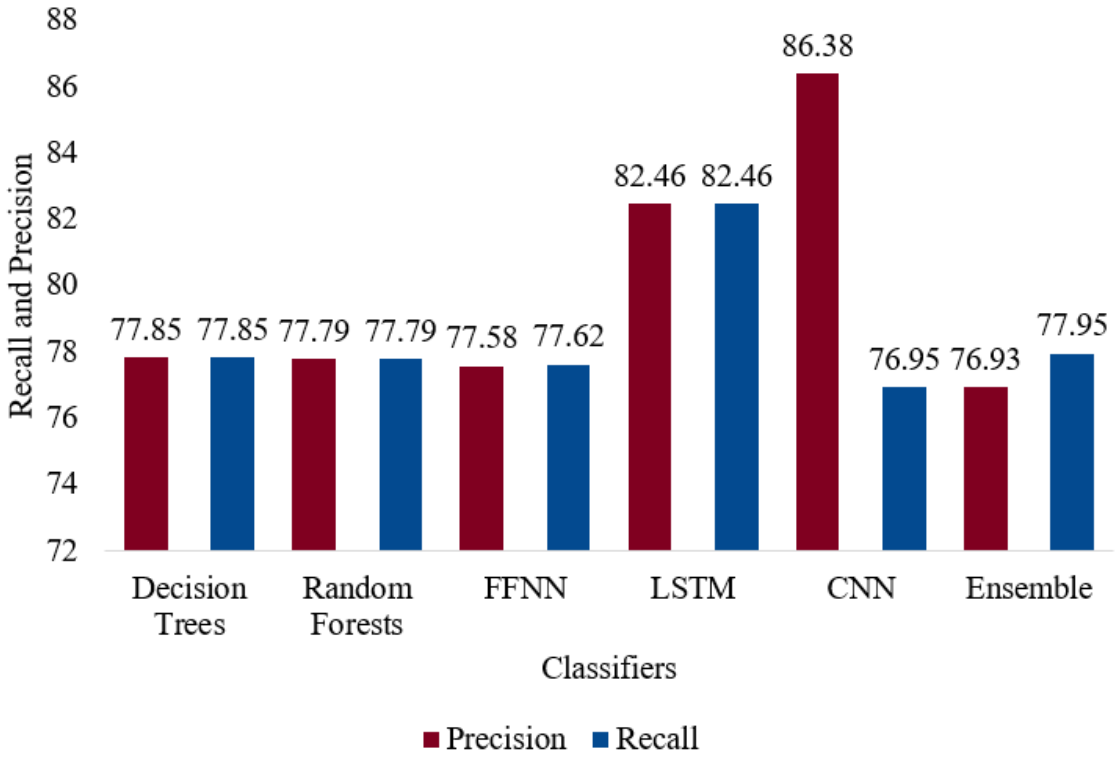


Figure 23: Small Dataset: Cyber-persona Recall and Precision after Up-Sampling

Large Dataset. In this section we present the results of our cyber-persona identification on the large dataset. the accuracy and F_1 score are presented in Table 7, while the recall and precision are presented in Fig 24 and Fig. 25. The loss and accuracy of the FFNN model are illustrated in Fig. 18 and Fig. 19, respectively. The results show that the ensemble provides the best performance before data up-sampling, with an accuracy equal to 98.85%, and F_1 score equal to 98.88%. The ensemble gave the best performance after up-sampling the data, with an accuracy equal to 98.88% and F_1 score equal to 98.88%. Moreover, Fig. 29 and Fig. 27 present the results obtained during the grid search conducted on the large dataset by considering different flow window sizes in the case of cyber-persona identification. Table 9 shows the flow window sizes providing the best results. As can be seen, considering one flow with embeddings provides the best results before and after up-sampling. Decision tree provides the best performance before up-sampling, with an accuracy equal to 98.41% and

F_1 equal to 98.38%. It also provides the best performance after up-sampling, with accuracy and F_1 equal to 98.77%.

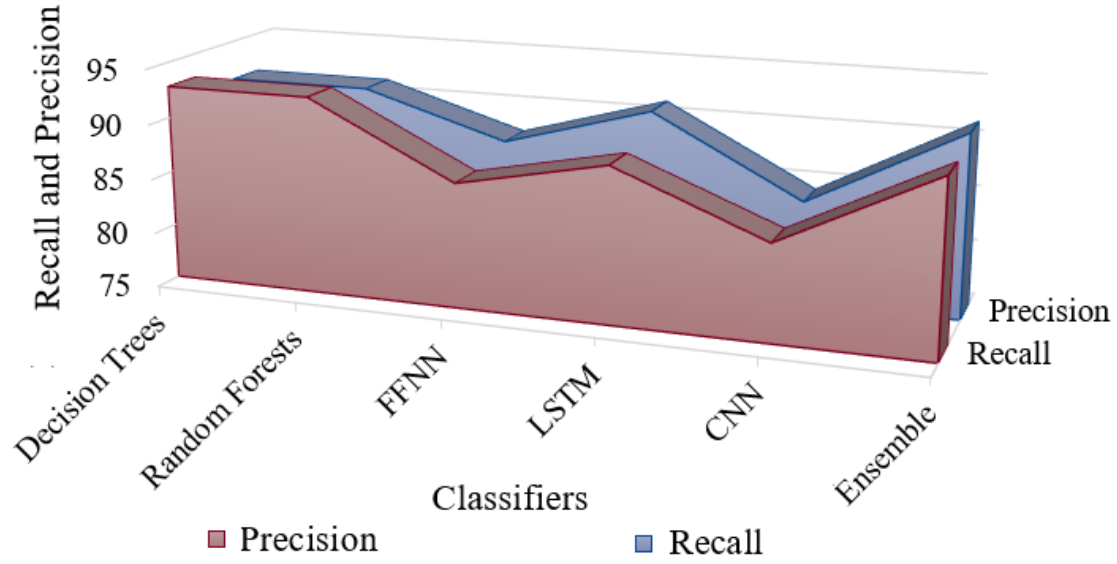


Figure 24: Large Dataset: Cyber-persona Recall and Precision before Up-Sampling

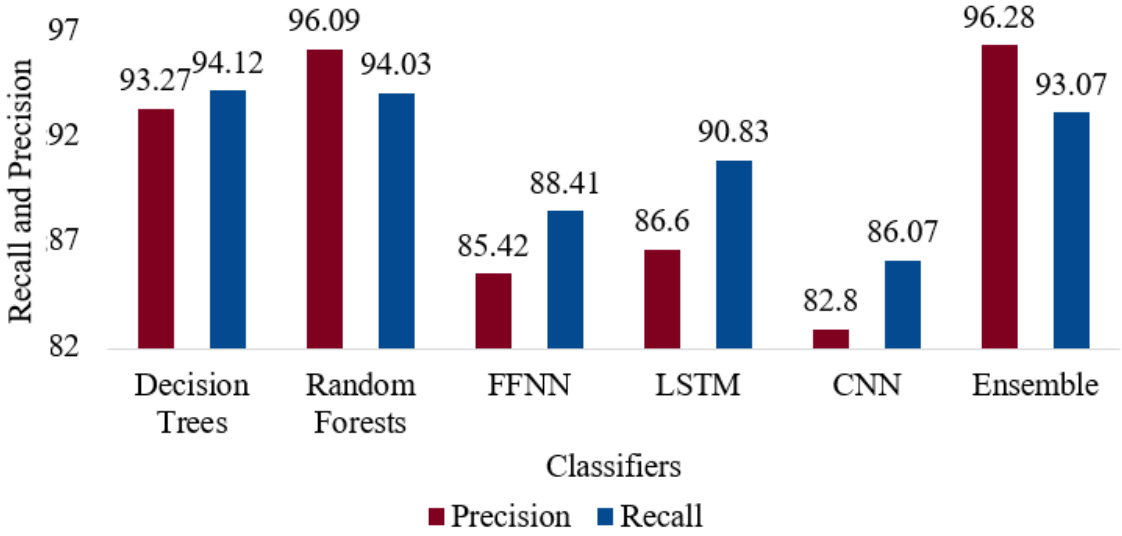


Figure 25: Large Dataset: Cyber-persona Recall and Precision after Up-Sampling

4.5 Insider Threat Detection Results

We test our insider threat detection approach on the small data, using the FFNN model trained and presented in Section 4.4. We experimentally derive a threshold value equal to 0.788 (see Fig. 26), whereby all predictions with a probability below this threshold will be flagged as suspicious. We obtain an AUC equal to 92.91%.

On the other hand, we have applied three outlier detection models on the large dataset to detect behavioural deviations. Fig. 28a, Fig. 28b and Fig. 28c illustrate respectively the results provided by our insider threat detection module in the case of cyber-persona type victim of an insider threat incident. The mentioned figures illustrate the network flows flagged as abnormal in red while normal ones are in green. Notice that the data dimensionality was reduced to two dimensions for the visualization, using *t-sne*[7].

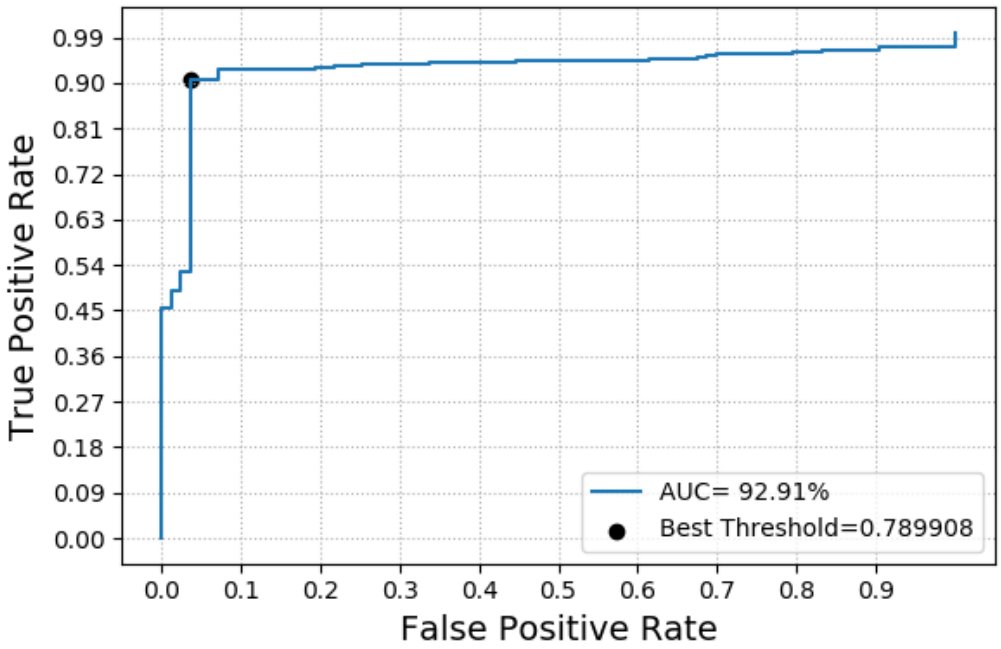


Figure 26: Small Dataset: Insider Threat Detectio ROC Curve

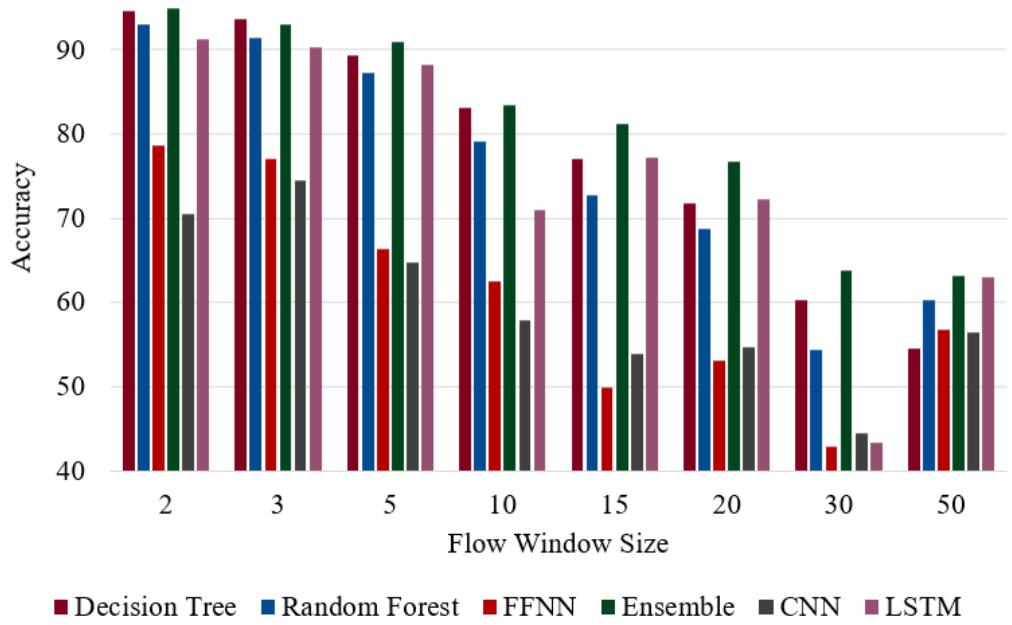
[7] `https://scikit-learn.org/`, accessed on November 15, 2020

4.6 Results Analysis and Discussion

This section compares the results obtained by considering single flows or windows of flows with embedding and with/without data up-sampling.

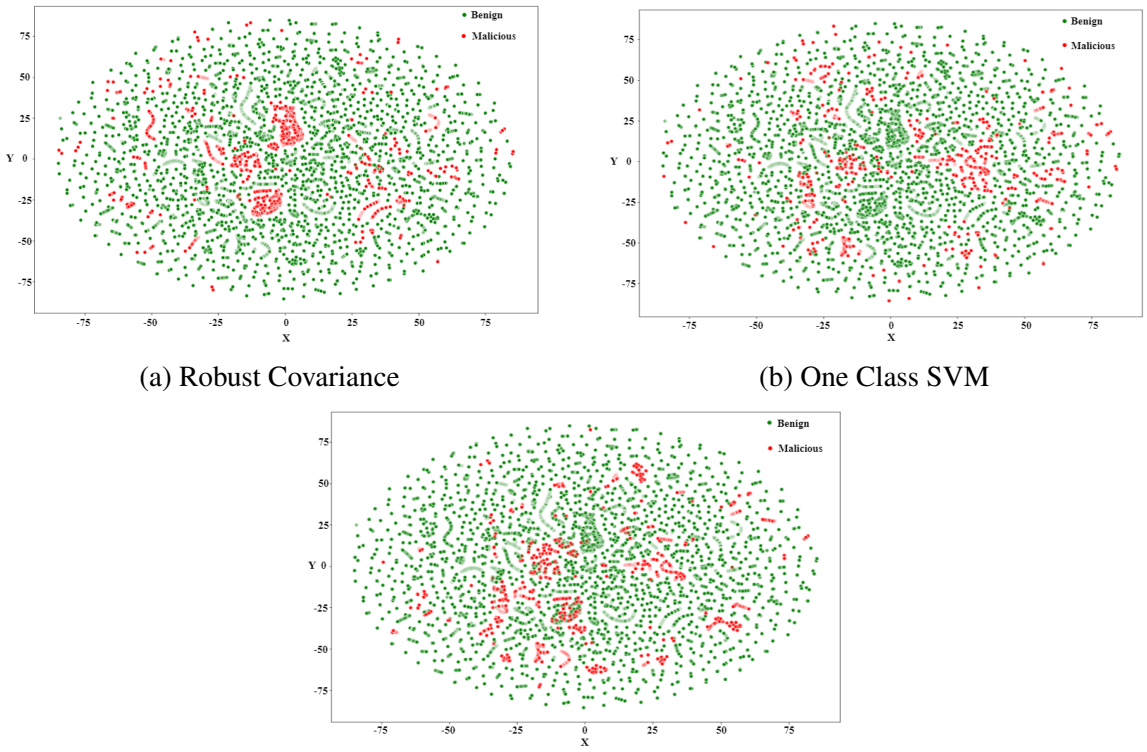
4.6.1 User Profiling

The experiments conducted indicate the following: in the case of the small dataset, when considering single flows and up-sampling the data, the best results are obtained by the ensemble of models (75.82% accuracy and 75.82% F_1 score). In contrast, when considering the flow window size of 30 and up-sampling the data, the best results are achieved by random forests (93.65% accuracy and 93.46% F_1 score). However, for the large dataset, single flows and up-sampling provide the best performance results using the ensemble model with 98.89% of accuracy and 98.89% of F_1 score.



(a) Accuracy per Different Flow Windows Sizes

Figure 27: Large Dataset: Cyber-persona's Accuracy Grid Search after Up-Sampling



(a) Robust Covariance

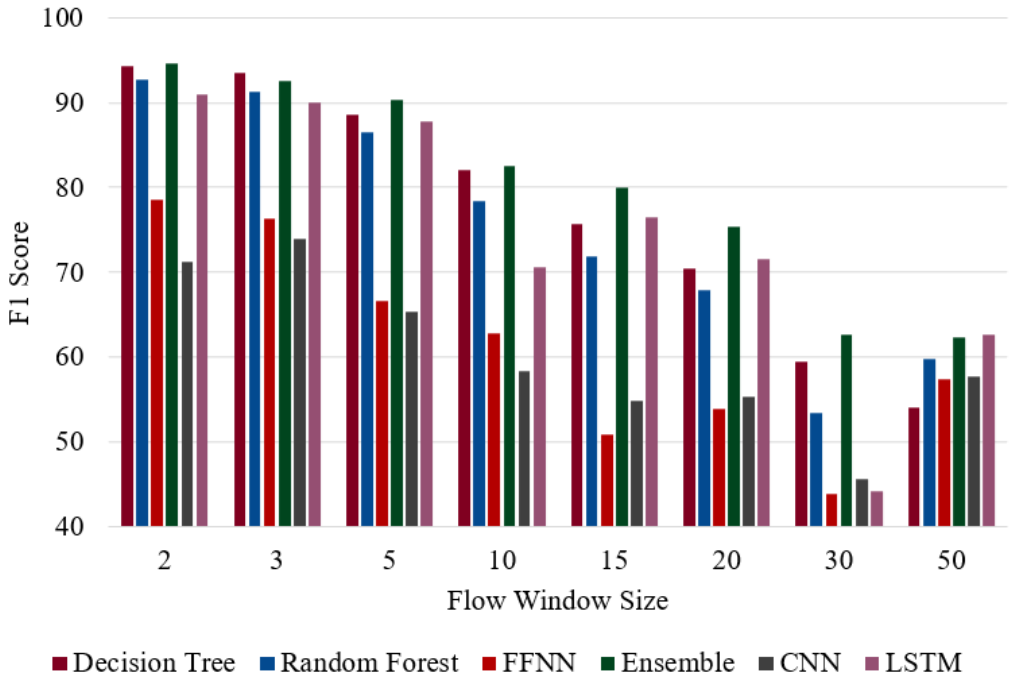
(b) One Class SVM

(c) Isolation Forests

Figure 28: Example of Insider Threat Detection on the Large Dataset

4.6.2 Cyber-persona Identification

The experiments conducted indicate the following: In the small dataset, considering a flow window size of 30 and data up-sampling provides the best results for the FFNN model (98.56% accuracy and 98.54% F_1 score). In contrast, when considering a flow window size of 50 without up-sampling, the best results are achieved by random forest (96.36% accuracy and 95.46% F_1 score). In the case of the large dataset, when considering single flows, the best results are obtained by the ensemble (98.85% for accuracy and 98.88 forF_1 score) before up-sampling and 98.91% for both accuracy and F_1 score, after up-sampling.



(a) F_1 Score per Different Flow Windows Sizes

Figure 29: Large Dataset: Cyber-persona's F1 Score Grid Search after Up-Sampling

4.7 Visualization

It is essential to visualize our approach's results in a dashboard to extract the actionable threat intelligence on time. To this end, we insert the network flows and their corresponding predictions (the most important ones are insider's *Id*, *cyber-persona*and *alerts*) into an *elasticsearch* index and visualizes them using *kibana* dashboards.

4.8 Conclusion

In this chapter, we have presented our experimental evaluation. We first presented our experimental setup. Second, we have presented the results obtained using the different modules (machine/human segregation, user profiling, cyber-persona identification, and insider threat detection). Third, we have provided a comparison between the different employed

models. Finally, we have presented our dashboard. In the next chapter, we will present this thesis's conclusions and the future work of this thesis.

Chapter 5

Conclusion and Future Work

Insider threat detection has become a relevant research topic due to the dramatic increase in its frequency and financial impact on targeted organizations. Cybercriminals take advantage of the careless insiders to launch cyberattacks from inside the organizations, making the firewalls and sophisticated techniques to prevent attacks from the outside insufficient. This thesis presented an automatic framework for analyzing the insiders' behaviour in the organizations' network. We designed and implemented a network traffic-based user profiling, cyber-persona identification, and abnormal behaviour detection system in this context. The system leverages supervised machine learning and deep learning techniques (including decision trees, random forests and deep feed-forward neural networks, LSTMs and CNNs) and outlier detection techniques (including isolation forests, one-class SVM and robust covariance).

When receiving streams of network flows, our framework first segregates the flows generated by machines (e.g., routers, printers) using a binary classifier. After that, we feed each network flow to the user profiling and the cyber-persona identification modules. Those two modules detect the *Id* and the *cyber-persona* type of the users by analyzing their network flows using supervised machine learning algorithms. Finally, we feed the network flows to the insider threat detection module, which employs an outlier detection model to detect

user's behavioural deviations. Our framework detects any behavioural deviation from the user's normal behaviour or the user's cyber-persona type's normal behaviour. We applied our approach to a real-world dataset provided by our partners and conducted an experimental study, which involved training different machine learning models. We used labelled network traffic data for training the models to provide the capability to timely detect the behavioural deviations of users with respect to their profiles and cyber-persona types. Our results demonstrated that our framework has a high performance, as demonstrated by our extensive experimental study. Corporate and academic organizations can benefit from deploying the presented framework to detect insider threats in near-real-time. In future work, we aim to investigate further the use of other machine learning and deep learning models. We will also apply our approach to large datasets from different sources.

Bibliography

[1] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[2] T. Bakhshi and B. Ghita. Openflow-enabled user traffic profiling in campus software defined networks. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2016.

[3] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[4] S. M. Bellovin. The insider attack problem nature and scope. In *Insider Attack and Cyber Security*, pages 1–4. Springer, 2008.

[5] M. Bishop. The insider problem revisited. In *Proceedings of the 2005 workshop on New security paradigms*, pages 75–76. ACM, 2005.

[6] M. Bishop, H. M. Conboy, H. Phan, B. I. Simidchieva, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, and S. Peisert. Insider threat identification by process analysis. In *2014 IEEE Security and Privacy Workshops*, pages 251–264. IEEE, 2014.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[9] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[11] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions on Information Forensics and Security*, 11(1):114–125, 2015.

[12] A. Cooper et al. *The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity*, volume 2. Sams Indianapolis, 2004.

[13] Deyan Georgie. 20 Insider Threat Statistics to Look out for in 2020, 2020.

[14] G. Doss and G. Tejay. Developing insider attack detection model: a grounded approach. In *2009 IEEE International Conference on Intelligence and Security Informatics*, pages 107–112. IEEE, 2009.

[15] N. Elmrabit, S.-H. Yang, L. Yang, and H. Zhou. Insider threat risk prediction based on bayesian network. *Computers & Security*, page 101908, 2020.

[16] P. Ferreira, D. C. Le, and N. Zincir-Heywood. Exploring feature normalization and temporal information for machine learning based insider threat detection. In *2019 15th International Conference on Network and Service Management (CNSM)*, pages 1–7. IEEE, 2019.

[17] Y. Gao, J. Tao, L. Zeng, X. Fang, Q. Fang, and X. Li. User profiling with campus wi-fi access trace and network traffic. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 922–927. IEEE, 2019.

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

[19] D. Godoy and A. Amandi. User profiling in personal information agents: a survey. *The Knowledge Engineering Review*, 20(4):329, 2005.

[20] M. Gratian, D. Bhansali, M. Cukier, and J. Dykstra. Identifying infected users via network traffic. *Computers & Security*, 80:306–316, 2019.

[21] A. Graves. Long short-term memory. In *Supervised sequence labelling with recurrent neural networks*, pages 37–45. Springer, 2012.

[22] F. L. Greitzer, A. P. Moore, D. M. Cappelli, D. H. Andrews, L. A. Carroll, and T. D. Hull. Combating the insider cyber threat. *IEEE Security & Privacy*, 6(1):61–64, 2008.

[23] J. Happa et al. Insider-threat detection using gaussian mixture models and sensitivity profiles. *Computers & Security*, 77:838–859, 2018.

[24] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[25] J. Hunker and C. W. Probst. Insiders and insider threats-an overview of definitions and mitigation techniques. *JoWUA*, 2(1):4–27, 2011.

[26] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis. An insider threat prediction model. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 26–37. Springer, 2010.

[27] J. Kim, M. Park, H. Kim, S. Cho, and P. Kang. Insider threat detection based on user behavior modeling and anomaly detection algorithms. *Applied Sciences*, 9(19):4018, 2019.

[28] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein. *Logistic regression*. Springer, 2002.

[29] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[30] D. C. Le and A. N. Zincir-Heywood. Machine learning based insider threat modelling and detection. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1–6. IEEE, 2019.

[31] D. C. Le and N. Zincir-Heywood. Exploring anomalous behaviour detection and classification for insider threat identification. *International Journal of Network Management*, page e2109, 2020.

[32] D. C. Le, N. Zincir-Heywood, and M. I. Heywood. Analyzing data granularity levels for insider threat detection using machine learning. *IEEE Transactions on Network and Service Management*, 17(1):30–44, 2020.

[33] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[34] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[35] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[36] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[37] L. Liu, C. Chen, J. Zhang, O. De Vel, and Y. Xiang. Unsupervised insider detection through neural feature learning and model optimisation. In *International Conference on Network and System Security*, pages 18–36. Springer, 2019.

[38] L. Liu, O. De Vel, C. Chen, J. Zhang, and Y. Xiang. Anomaly-based insider threat detection using deep autoencoders. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 39–48. IEEE, 2018.

[39] O. Lo, W. J. Buchanan, P. Griffiths, and R. Macfarlane. Distance measurement methods for improved insider threat detection. *Security and Communication Networks*, 2018(5906368):1–18, 2018.

[40] J. Lu and R. K. Wong. Insider threat detection with long short-term memory. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–10, 2019.

[41] G. Magklaras, S. Furnell, and P. J. Brooke. Towards an insider threat prediction specification language. *Information management & computer security*, 14(4):361–381, 2006.

[42] M. A. Maloof and G. D. Stephens. Elicit: A system for detecting insiders who violate need-to-know. In *International Workshop on Recent Advances in Intrusion Detection*, pages 146–166. Springer, 2007.

[43] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[44] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[45] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*, volume 821. John Wiley & Sons, 2012.

[46] A. Moore. K-means and hierarchical clustering, 2001.

[47] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[48] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 11 user fingerprinting. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 99–110, 2007.

[49] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli. Autoencoders. In *Machine learning*, pages 193–208. Elsevier, 2020.

[50] D. M. POWERS. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2011.

[51] J. R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[52] T. Rashid, I. Agrafiotis, and J. R. Nurse. A new take on detecting insider threats: exploring the use of hidden markov models. In *Proceedings of the 8th ACM CCS International workshop on managing insider security threats*, pages 47–56, 2016.

[53] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, pages 41–46, 2001.

[54] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.

[55] S. Schiaffino and A. Amandi. Intelligent user profiling. In *Artificial Intelligence An International Perspective*, pages 193–216. Springer, 2009.

[56] A. Senior. A hidden markov model fingerprint classifier. In *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No. 97CB36136)*, volume 1, pages 306–310. IEEE, 1997.

[57] F. Shaman, B. Ghita, N. Clarke, and A. Alruban. User profiling based on application-level using network metadata. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–8. IEEE, 2019.

[58] B. Sharma, P. Pokharel, and B. Joshi. User behavior analytics for anomaly detection using lstm autoencoder-insider threat detection. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–9, 2020.

[59] S. J. Sheather. Density estimation. *Statistical science*, pages 588–597, 2004.

[60] N. M. Sheykhkanloo and A. Hall. Insider threat detection using supervised machine learning algorithms on an extremely imbalanced dataset. *International Journal of Cyber Warfare and Terrorism (IJCWT)*, 10(2):1–26, 2020.

[61] C. Soh, S. Yu, A. Narayanan, S. Duraisamy, and L. Chen. Employee profiling via aspect-based sentiment and network for insider threats detection. *Expert Systems with Applications*, 135:351–361, 2019.

[62] S. Stathatos, A. Mishra, and C. A. Mattmann. Cyber persona identification via indirect feature analysis. *N/A*, 2018.

[63] K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera. Synergistic union of word2vec and lexicon for domain specific semantic similarity. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–6. IEEE, 2017.

[64] J. A. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[65] C. I. T. Team. Insider threat test dataset, 2016.

[66] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

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

[68] S. S. Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

[69] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[70] X. Ye, S.-S. Hong, and M.-M. Han. Feature engineering method using double-layer hidden markov model for insider threat detection. *International Journal of Fuzzy Logic and Intelligent Systems*, 20(1):17–25, 2020.

[71] Z. Yu, E. Xu, H. Du, B. Guo, and L. Yao. Inferring user profile attributes from multi-dimensional mobile phone sensory data. *IEEE Internet of Things Journal*, 6(3):5152–5162, 2019.

[72] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang. Insider threat detection with deep neural network. In *International Conference on Computational Science*, pages 43–54. Springer, 2018.

[73] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *IEEE communications surveys & tutorials*, 12(2):159–170, 2010.

[74] Z. Zhang, S. Wang, and G. Lu. An internal threat detection model based on denoising autoencoders. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 391–400. Springer, 2020.

[75] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.