INTRUSION DETECTION THROUGH UNSUPERVISED CONVENTIONAL MACHINE LEARNING MODELS

Nikolaos Mitsas
STUDENT NUMBER: 2090834

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. Grzegorz Chrupała
Dr. Paris Mavromoustakos Blom

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
January 2023

Acknowledgement

I would like to extend my appreciation to my supervisor, Dr. Grzegorz Chrupała, for his invaluable guidance, and expertise. I would also like to acknowledge the assistance provided by my university, Tilburg University, throughout the course of my studies, which has played a vital role in shaping my understanding of data science and fostering my intellectual growth. Finally, I am immensely grateful to my friends and family for their unwavering encouragement and support.

Word Count

8780

INTRUSION DETECTION THROUGH UNSUPERVISED CONVENTIONAL MACHINE LEARNING MODELS

MITSAS NIKOLAOS

This work focuses on investigating intrusion detection and developing an intrusion detection system (IDS). Intrusions refer to any illegal activities on computer systems or networks that aim to cause harm. Machine learning algorithms have recently become a common tool for reliable IDSs development. There has been a growing interest in intrusion detection due to its significant importance for the safety of network users. However, despite the existing research, detecting new intrusions types is difficult as they become more complex and advanced. Therefore, the thesis focuses on developing an IDS using unsupervised machine learning models, which can detect unknown intrusions by identifying hidden patterns in the data. Thus, the variation in the performance between a supervised machine learning approach in which Random Forest (RF) was used, and the three unsupervised machine learning models that used: Isolation Forest (IF), Elliptic Envelope (EE), and One-Class Support Vector Machine with Stochastic Gradient Descent and Nystroem kernel approximation (OCSVM-SGD-Nystroem) were compared and discussed. Additionally, the unsupervised models were compared with each other to determine which detects intrusion more effectively. In most of the research in the field, the models used have been trained and evaluated in outdated datasets. Recent studies use more complex and time-consuming models for intrusion detection, such as deep learning models. However, this study uses traditional machine learning models on an updated dataset, CICIDS2017, which contains a variety of new types of attacks. Previous studies on the dataset have focused on either supervised learning, leveraging the availability of full labels, or unsupervised intrusion detection using deep learning models. The supervised approach achieved high performance, while the unsupervised models exhibited relatively lower effectiveness but still exhibited promising results. Finally, OCSVM-SGD-Nystroem had the best accuracy among the unsupervised models. Despite the lower performance, unsupervised models can detect unseen intrusions and patterns that supervised models are unable.

1. Data Source/Code/Ethics Statement

The dataset was retrieved from the Canadian Institution for Cybersecurity (CIC), and there was no direct involvement in the collection of data from human or animal participants. The institution remains the owner of the respective dataset during and after the completion of the thesis. The author of the thesis acknowledges that they have no legal ownership of this dataset, and the code is publicly available on GitHub. The figures included in this work were created by the author.

2. Problem Statement and Research Goals

2.1 Scientific and societal relevance

The current advance and rapid development of technology and the Internet of Things (IoT) have created a great challenge for the protection of networks, servers, and their users, which have become vulnerable to many illegal activities or attacks, such as malicious behaviors or unauthorized access to servers or networks, to cause financial damages or compromise sensitive data, also known as intrusions, (Panigrahi and Borah 2018; Zhang et al. 2021). Despite the existence of firewalls and antivirus software, why is intrusion detection still a crucial aspect of network security? The answer to that question is simple: as technology is growing fast, intrusions also grow and become more complex and sophisticated, so they can still arise. Even though the increasing complexity of computer security measures, achieving complete security remains a challenge. Consequently, IDS development remains a crucial task for cybersecurity, (Kemmerer and Vigna 2002).

Recently, intrusion detection systems (IDSs) have become the most common approach for detecting intrusions, with many of them using machine learning algorithms. Intrusion detection is considered an anomaly detection process in the field of machine learning. IDSs based on anomaly detection are trying to find patterns in the data that deviate from the normal and identify them as anomalies, intrusions, (Data and Aritsugi 2021). As mentioned in the first paragraph, intrusions tend to happen more frequently and in advanced and new forms, making detection tasks more complex, (Modirrousta, Forghani, and Shoorehdeli 2022). Hence, unsupervised machine learning models are more suitable for real-world intrusion detection, as they possess the capability to detect unknown intrusions, which is critical in the context of emerging and evolving cyber-attacks. Unsupervised machine learning models do not rely on labels, meaning that they have no prior knowledge, ensuring that all intrusions are treated as unknown entities. However, they are trying to detect the intrusions through patterns in the data. This is particularly advantageous for intrusion detection as vast amounts of data must be handled, and labeling them can be expensive.

Despite the enormous number of studies in the field of intrusion detection, there are still some gaps in contributions in terms of new data and methods. Therefore, the novelty of this research in academic relevance aims to complement what has already been implemented. Conventional machine learning models have been applied in outdated datasets, while in the chosen dataset, which contains new and updated network flows and intrusions, almost all the studies in unsupervised learning have proposed deep learning models. The dataset is fully labeled, meaning that a significant portion of the research with it is based on supervised learning. Although it includes multiple types of intrusions, this study focuses exclusively on detecting whether a network flow represents an intrusion or a benign activity with unsupervised machine learning models. In this work, the labels were used only to evaluate the models.

2.2 Research questions

The central objective of the thesis is to develop an unsupervised machine learning method for the detection of intrusions in cybersecurity and to evaluate the performance of traditional machine learning models for such tasks. Three unsupervised machine learning models were used, namely, Isolation Forest (IF), Elliptic Envelope (EE), and One-Class Support Vector Machine with Stochastic Gradient Descent and Nystroem kernel approximation (OCSVM-SGD-Nystroem), because they are commonly used for unsupervised anomaly detection and utilize low computational cost, which was a critical consideration throughout the study. The first and primary research question addresses the effectiveness of those models in updated data since they have proven effective in outdated datasets.

- Can traditional unsupervised machine learning models effectively detect intrusions in updated data?

Two sub-questions were created, to help understand and assess the capabilities of the unsupervised approach in detecting intrusions. A supervised approach was first developed with Random Forest (RF) algorithm, to obtain a topline and compare it with the results of the unsupervised method. Consequently, the first research sub-question is:

- What is the variation in the performance between unsupervised learning and supervised learning methods for intrusion detection?

Finally, the second sub-question formulated is related to the performance of the unsupervised models used and the detection of the more efficient model among them.

- Which unsupervised algorithm among IF, EE, and OCSVM-SGD-Nystroem demonstrates better performance?

The dataset was cleaned, pre-processed, and some features with no discriminate power were excluded to ensure they were suitable for machine learning models. Next, the dataset was split into training, validation, and testing sets. The identical procedures were applied for both approaches, ensuring a thorough comparison between them, with the only difference in the supervised method, where RF was used with and without over-sampling the minority class since the dataset was imbalanced. Regarding answering the questions, and having an unbiased and comprehensive evaluation, the performance was defined as the result of the f1-score. In addition, the focus was also on the number of misclassified intrusions, which can have more severe consequences than false alarms. Further details regarding the models and the evaluation process are substantiated in Section 4.

2.3 Findings

The supervised approach achieved high results, with the RF algorithm demonstrating exceptional performance with and without over-sampling the minority class. Of course, as expected the results of the supervised learning were significantly higher than those of the unsupervised models. Among the unsupervised models, OCSVM-SGD-Nystroem achieved the highest accuracy regarding the f1-score, and EE managed to detect the most intrusions. IF demonstrated the worst performance, raising too many false alarms. In general, the unsupervised models generalized effectively to unseen data and identified a considerably large number of previously unseen intrusions.

The rest of the paper is organized as follows. Section 3 presents the literature for supervised and unsupervised intrusion detection with deep learning and traditional machine learning models. Section 4, introduces the CICIDS2017 dataset, the cleaning and pre-processing steps, the methods utilized, and the experimental setup. The results of the study and the comparison of the unsupervised models in terms of their respective performance, as well as with the topline, are presented in Section 5, and in Section 6, the answers to the research questions and discussion are displayed. Finally, Section 7 provides the conclusion of this work.

3. Related Work

The present section provides a brief historical overview of the literature relevant to this thesis, highlighting the gaps and limitations of the current literature. The methods utilized in this thesis are introduced to alternative approaches. Firstly, the wider research field concerning supervised intrusion detection is presented. After that, the focus shifts to similar unsupervised methods with traditional machine learning models in outdated datasets and approaches with deep learning models in more recent datasets.

3.1 A brief historic overview for intrusion detection

Over the last decades, there have been more and more studies in the field of intrusion detection for the creation of automatic techniques to detect them. That need originated with the fast and growing amount of data and technology. Nowadays with everyone having access to computer networks, the number of network flows and intrusions has grown exponentially. Originally intrusion detection heavily relied on human intervention to identify the attacks, which looks impossible now, (Sinclair, Pierce, and Matzner 1999). The first attempts for automatic IDSs based on machine learning started around two decades ago. The form of intrusion can be described as anomaly detection, since most of the network flows are benign, and the systems are trying to flag the ones that deviate from the normal throughout patterns in the data, (Lee, Stolfo, and Mok 1999).

After all that time and despite the enormous number of studies conducted in the field, the development of reliable and comprehensive IDSs is still considered a challenging task. The basis for this derives from the increasing number of complex and new attacks that occur, which illustrates distinct anomalous behaviors. Therefore, intrusion detection methods require techniques that employ automated and intelligent strategies for network intrusion detection to cope with them.

3.2 Supervised anomaly detection

Supervised machine learning approaches have been widely explored for network intrusion detection, using both outdated and updated datasets, and have shown significant performance. In supervised learning, models are trained with labels to identify network flows that represent intrusions. In both studies by Ahmad et al. (2021) and Ravipati and Abualkibash (2019), traditional machine learning models demonstrate that they can achieve high accuracy in detecting intrusion in both binary and multi-class anomaly detection tasks. Specifically, in Ahmad's study, anomalies were treated in both binary and multi-class classification tasks, with the former being more similar to the approach in this thesis. From the results of the binary classification, among all the models that were used, including a deep learning model, the RF outperformed all of them, with an accuracy of 98.8%. Furthermore, the Support Vector Machine (SVM) also performed better than the Artificial Neural Network (ANN), with an accuracy of 95.67% while ANN achieved an accuracy of 91.7%. These findings suggest that traditional machine learning models can outperform deep learning models in supervised intrusion detection. In the multi-class classification task for intrusion detection in both approaches mentioned above, RF outperforms all the other models used.

Another supervised approach, using the CICIDS2017 dataset, which contains updated data, Sharafaldin, Lashkari, and Ghorbani (2018) aimed to detect each of the 14 intrusions. RF, k-Nearest Neighbors (K-NN), and Iterative Dichotomiser 3 (ID3) demonstrated the highest performance, with RF being computationally more efficient than the other two, achieving an accuracy of 98%. Although the Multilayer Perceptron (MLP) was applied, it exhibited poor performance, approximately 20% lower than the RF model, and required additional processing time. In general, systems can detect intrusion with high precision in supervised intrusion detection. However, there are drawbacks to these approaches, which is why the thesis focused on unsupervised machine learning. The supervised models that are fully trained with labels may not generalize well in detecting intrusions that do not exist in the dataset that they were trained on, (Data and Aritsugi 2021). RF algorithm has been found to outperform a diversity of other algorithms in supervised intrusion detection and therefore it was utilized for the supervised approach in this thesis.

3.3 Unsupervised anomaly detection

Unlike supervised methods, unsupervised machine learning models do not rely on labeled data. Labeling a dataset is often a time-consuming and expensive process. It requires human experts to manually label each instance in the dataset. Additionally, supervised models have limitations in detecting new or previously unseen types of attacks. In a study by Leung and Leckie (2005), the researchers introduce a novel clustering approach with density-based and grid-based clusters for unsupervised intrusion detection and evaluated them on the KDD-Cup dataset. Although KDD-Cup contains real-world data and is commonly used for intrusion detection, it was published in 1999, meaning that the network traffic and intrusions of the dataset are outdated. The proposed clustering approaches were designed to handle high dimensional and large datasets and were found to have similar results to commonly used anomaly detection algorithms, such as OC-SVM, but required significantly less computational time. In a broad sense, clustering is a common technique used for anomaly detection, where the models create clusters, and the instances that belong to these clusters are identified as benign points, while the minority that falls outside the clusters is classified as anomalies. Unsupervised classification machine learning models such as IF and OC-SVM, have also shown promising results for intrusion detection. For example, in the research contacted by Zhang and Zulkernine (2006) using the same dataset, OC-SVM was used along with other clustering models and (K-NN), which can also be used as an unsupervised anomaly detection algorithm. In this research, K-NN did not perform well, while OC-SVM had promising results that were similar to the clustering algorithm. In both studies, different types of unsupervised traditional machine learning algorithms were used with promising results, demonstrating that conventional machine learning models can efficiently detect intrusions. However, a major disadvantage of these models is that they were trained and evaluated in out-of-date datasets that are not representative of current network flows and intrusions, making them unable to detect new types of intrusions.

As already mentioned, most of the new research in unsupervised network intrusion detection tends to develop systems with deep learning models. One of the most frequent deep learning models used for intrusion detection is Self-Organized Maps (SOM), (Barletta et al. 2020; Qu et al. 2021; Chen et al. 2021). SOM is widely used in combination with other models, like in Barletta et al. (2020) work, where they proposed a hybrid model which combined SOM with a k-means clustering algorithm. Concerning the simple SOM, it resulted in more accurate results. The combination of these two algorithms allowed the authors to identify clusters of data points that were potential intrusions while preserving the topology of the original data. Another such case was in Chen et al. (2021) research, where two deep learning models, namely, SOM and Deep Auto-encoding Gaussian Mixture Model (DAGMM), were merged. The SOM algorithm extracted features that preserved the space topology, to reduce the dimension of the dataset because DAGMM performs better in low dimensional data. The new model achieved a high value of accuracy in both datasets, with 89% accuracy in the first dataset and 93% accuracy in the second dataset, outperforming the original DAGMM algorithm by approximately 12% and 3% in each dataset.

A broader approach to the CICIDS2017 dataset was taken by Belarbi et al. (2022), in which the researchers proposed Deep Belief Networks (DBNs) for unsupervised intrusion detection in a multi-class environment. DBNs were used because they are known for their ability to effectively detect anomalies and learn patterns in high-dimensional data. Moreover, they also developed a Multi-Layer Perceptron (MLP) model, which was evaluated alongside 6 other state-of-the-art deep learning models in the same dataset. Some of these models were evaluated for binary intrusion detection. The proposed DBNs model outperformed the MLP and some of the state-of-the-art models, even though their focus was binary, achieving an accuracy of 94% and MLP 87.3%. In conclusion, these studies suggest that deep learning models perform effectively for unsupervised intrusion detection, but they are more complex and computationally expensive than traditional machine learning models.

Traditional machine learning models yielded promising results in outdated datasets, which is why they were used for this thesis. The chosen models are commonly used for intrusion detection, as well as in a variety of other types of anomaly detection. For instance, in the study by Poon et al. (2021), IF and OC-SVM were utilized on a dataset for data quality control, demonstrating promising results for unsupervised anomaly detection with an f1-score of 69.2% and 72.0%, respectively. They were compared with a variety of other models, such as clusters that achieved slightly better results, approximately 73.2%. However, they were excluded from this work because they are computationally expensive. EE is also used for unsupervised anomaly detection, but not as frequently as the previous ones. In Almuhtaram, Zamyadi, and Hofmann (2021) study, EE had the same f1-score with OC-SVM and marginally outperform IF, highlighting its effectiveness and showing that it has the potential as a valuable model for unsupervised anomaly detection. In addition, OC-SVM is computationally expensive since its complexity requires a quadratic time of the number of samples. Due to that and based on the documentation in scikit-learn, (Pedregosa et al. 2011), where was cited that OC-SVM with Stochastic Gradient Descend is more suited for large datasets since it speeds up the time of the model it was used in this work. Kernel approximation was used with this model because that combination can obtain results similar to the OC-SVM. Williams and Seeger (2000) and Yang et al. (2012) introduced Nystrom kernel approximation, which makes an accurate approximation and reduces the time and memory required. This was justified, by comparing it with other kernel approximation methods, such as Random Fourier Features.

4. Methodology

In this section, all components of the experiment of this thesis are described. It begins by outlining the data source and the techniques used to clean and pre-process the data for the algorithms. The topline supervised model is then explained in technical terms, followed by a formal specification of the three unsupervised models' architecture. The section proceeds with a detailed explanation of the experimental setup and the evaluation metrics. The evaluation framework is explained regarding the research questions. Finally, the hardware and software infrastructure for estimation purposes is presented.

4.1 Data and pre-processing

The dataset used is the "Intrusion Detection Evaluation Dataset (CICIDS2017)" and was retrieved from the Canadian Institute for Cybersecurity, (Canadian Institute for Cybersecurity CIC). The CICIDS2017 dataset in comparison with previous datasets is updated with current network flows and intrusions. According to Gharib et al. (2016) research ensuring the reliability and accuracy of benchmark datasets is crucial for intrusion detection and 11 criteria have been formulated for that reason. The chosen dataset is the first public dataset for intrusion detection that satisfy all those criteria. The dataset was created in a five-day experiment, which lasted from Monday morning, July 3, 2017, until Friday afternoon, July 7, 2017, and captured both benign and intrusion network flows in a realistic, simulated environment. It is automatically fully labeled, with manual intervention by security experts to ensure the accuracy of the intrusions. The different types of intrusions are identified based on the data packets that move from and to the destination port. Furthermore, CICIDS2017 contains 2,830,743 instances, 78 numerical features, and a column with labels. The dataset is class imbalanced, with an imbalance ratio of approximately 20%, where attacks are the minority class.

The focus of this work was to develop a binary classification system that can detect whether a network flow is benign or a potential intrusion. A major challenge encountered during the research was the computational time required for the models to run efficiently. To address this issue, the dataset size was reduced by excluding the data from the last day of the experiment, reducing it to 2,127,498 instances, around 25% smaller, and 11 types of intrusions, while still consisting of a diverse range of attack scenarios and a new class imbalance ratio of approximately 11%. It should be emphasized that the excluded data only contained benign flows and the three types of intrusions that were left out, meaning that all the information from the remaining intrusions was not affected.

After combining the data from the first four days into a dataset, and to ensure that the dataset was suitable for the machine learning models, data cleaning and preprocessing steps took place. During exploratory data analysis, it was observed that the dataset contained 307 missing values in just one column, namely "Flow bytes/s", and 1,929 infinite values, with 811 located in the same column and 1,118 in the "Flow Packets/s" column. All the other columns were cleaned from missing and infinite values but contained outliers. In addition, out of the 2,127,498 rows, 201,442 duplicates rows were identified, one column appeared twice, "Fwd Header Length" and "Fwd Header Length.1", and seven other columns had only 0 values.

The initial action taken to clean the dataset was to drop the duplicate rows, keeping the first instance that appeared from the duplicates to avoid them from biasing the models' performance. Furthermore, the decision to remove the duplicated data was justified based on the findings of Lanvin et al. (2022), who demonstrated that these duplicates were generated as a result of issues from the CICFlowMeter, a network traffic analysis tool, used for the creation of the dataset, and could consequently lead to misleading results. This action was implemented first to ensure that the mean of the columns was not biased from the duplicate instances, as it was used to impute the missing values, which was the next step. Next, the infinite values were addressed by replacing them with the maximum value of the corresponding column multiplied by two, based on the rationale that infinite values typically represent enormous values, which can be approximated in this way. In addition, those values were considered outliers. In general, the outliers found in the dataset were not addressed since they can provide valuable insights for anomaly detection tasks.

For the pre-processing, first, the column of the labels, which contained the categorical values of the benign network flows and 11 different types of intrusions, was replaced, with 0 for the benign instances and 1 for all the attacks, to make it binary. For the feature selection, the feature that appeared twice, and the other seven that contained 0 values were excluded. After these steps, the dataset had 1,926,056 instances, 69 features, and the column with the labels. Afterward, it was split into training, validation, and test sets by 80%, 10%, and 10%, respectively. The splitting process did not employ stratification, and the instances were shuffled to simulate real-world conditions for the unsupervised approach, considering that the labels were unknown and, consequently, should not be used to ensure equal distribution during the split. Finally, the data were scaled with the standard scaler, which is not robust to outliers and does not influence them, since they can give insights into the anomaly instances, as mentioned above. Appendix 1 illustrates more details about the dataset.

4.2 Topline supervised model

RF classifier was the model used for the topline-supervised approach. RF is a powerful supervised machine learning algorithm widely used for anomaly detection and other fields, and it was introduced by Breiman (2001). The algorithm gets its name because it combines the predictions of multiple decision trees through majority voting to make a final classification decision, meaning that the final prediction is the class that has been more frequently predicted among the decision trees developed.

$$y_{\text{pred}} = \text{mode}\{C_1(x), C_2(x), \ldots, C_m(x)\} \tag{1}$$

Formula 1 shows how the final prediction is found, where y_{pred} is the final prediction of the majority voting. C represents each of the m classifiers used, in the case of RF each classifier is a decision tree, C(x) is the prediction of each classifier, and x is the instance tested. The mode function returns the most frequent prediction.

During training, the RF algorithm uses bootstrap aggregating or bagging to create random samples of the training dataset for each decision tree. Bagging repeatedly draws random samples from the training set with replacement and for the prediction of new data, it takes the majority vote of the decision trees that have been constructed. RF also applies feature bagging during training, selecting random features with replacements to avoid choosing variables that are too correlated with the class labels because if the decision trees are related, they will generate the same errors and make similar predictions. Due to that, the algorithm develops a diverse set of decision trees, by combining the prediction of those trees, it creates a more robust and accurate model, while also by using different samples and features in each decision tree, RF can detect patterns between the data and reduce over-fitting. However, not all samples and features are used to grow the trees in the training face. The ones that were not used are called out-of-bag (OOB) and are used to estimate the algorithm's performance. In conclusion, another crucial advance of RF is that it can give insights about the importance of the features in two ways, by computing the feature importance scores based on the decrease in the OOB error rate when a particular feature is permuted or by calculating the division of the sum of the normalized feature importance of each decision tree with the total number of trees,

$$RFf_i(i) = \frac{1}{T} \sum_{j \in alltrees} \text{norm} f(ij) \tag{2}$$

where RFfi(i) is the feature importance of the i feature calculated from all trees in the RF model, normfi(ij) is the normalized feature importance for the i feature in the j tree and T is the total number of trees.

4.3 Unsupervised machine learning approach

4.3.1 Isolation Forest. The IF algorithm, introduced by Liu, Ting, and Zhou (2008), proposes a fundamentally distinct approach for anomaly detection. Unlike previous models, IF does not rely on a prior description of benign points or explicit designation of anomalies. Instead, it leverages anomalies by employing an ensemble of decision trees, similar to the Random Forest (RF) algorithm. However, in the context of IF, the decision trees operate in a district manner partitioning the data randomly instead of striving for purity with measures like entropy or gini. Each tree in the IF is constructed by recursively partitioning the data, with the primary aim of the algorithm to isolate anomalies with as few splits as possible. The anomalies tend to have distinct values that deviate from the norm, making them easier separably. Therefore, they require fewer partitions to be isolated. Detecting benign points, on the other hand, requires more splits, resulting in larger trees. The number of splits required to isolate a point serves as the basis to generate the anomaly score, which can also be referred to as a path. A shorter path indicates a higher likelihood of being an anomaly. This process is repeated multiple times, utilizing different decision trees with different attributes and sub-samples for each instance, and an average anomaly score is computed. In that way, an average score is assigned to each instance to ensure a more robust evaluation. If this average score is close to 1, the instance is more likely to be an anomaly, while a score closer to 0.5 suggests a benign point.

One of the main advantages of IF is that it is computationally fast and efficient even with large and high-dimension datasets. It can also handle global and local anomalies, making it a versatile choice for various anomaly detection tasks. Furthermore, IF requires relatively less memory conversely to other unsupervised anomaly detection models.

4.3.2 Elliptic envelope. EE is an unsupervised machine learning algorithm commonly used for anomaly detection. Its advantages are that it is robust to noise, works well in high-dimensional data, and does not require a long time for processing. However, the performance of the algorithm may suffer if the underlying data distribution is not elliptical or if there are multiple clusters in the data. In other words, the assumptions underlying the EE algorithm may not always hold in real-world datasets. The EE algorithm develops a cluster based on the distance from the center of the data distribution, which can also be called an ellipse. It attempts to fit most of the instances inside the ellipse, where all the points that fall inside the ellipse are identified as normal points, while the ones that fall far from it are identified as anomaly points. For a robust estimation of the size and shape of the ellipse, EE uses a statistical measure called FAST-minimum covariance determinant (FAST-MCD), (Rousseeuw and Driessen 1999). FAST-MCD is a statistical measure that gives a robust estimation of the covariance matrix, taking into account the presence of outliers. The metric used to calculate the distance between the point and the center of the ellipse is computed with the Mahalanobis distance, which also considers the covariance of the data and derives a measure of outlyingness. By using the contamination parameter, which specifies the expected proportion of anomalies in the dataset, the EE algorithm determines the threshold to detect the anomalies, (Ashrafuzzaman et al. 2020).

4.3.3 One Class Support Vector Machine with Stochastic Gradient Descend and Nystroem kernel approximation. OC-SVM is one of the most common unsupervised anomaly detection techniques used to distinguish instances of a particular class from another, in that context, anomalous data. What makes this algorithm so powerful is that, with the use of kernel approximation techniques, it can project the data into a higher dimensional space, making the points more separable, also known as the kernel trick. The kernel trick can turn data that were not separable linearly in the original space, separable with a straight line called a hyperplane in the higher dimension space. After the separation and when we project back to the former dimensions, a round boundary has created to separate the data. As a result, in the OC-SVM algorithm, the benign points are enclosed by the hypersphere defined by the hyperplane, while the points outside are identified as anomalies. According to Schölkopf et al. (1999), the algorithm is trying to minimize the hypersphere by maximizing the margin that encloses all the benign points in the higher-dimension space. The margin in OC-SVM refers to the distance between the hypersphere and the closest normal points.

However, a major drawback of this algorithm is the quadratic complexity that requires a long processing time. Therefore, in this thesis, OCSVM-SGD-Nystroem was used. This combination can yield similar results to the original OC-SVM in less time since in OCSVM-SGD-Nystroem, the model's optimization parameters are updated incrementally in a subset of the data, as opposed to the OC-SVM that processes all the data together. Finally, the Nystroem kernel approximation was used alongside OC-SVM with Stochastic Gradient Descend, as it is also recommended in sci-kit-learn, to achieve the kernel trick. Nystroem does not compute the kernel function on the entire dataset, but it uses a small subset of the data, making it efficient for large datasets without decreasing the accuracy, (Williams and Seeger 2000).

4.4 Experimental setup

After the cleaning and pre-processing, the dataset was ready to implement the machine learning models. Grid Search was employed for models that had a limited number of parameters for the hyperparameter tuning because it tests all possible parameter combinations and can find the best among them. Conversely, Random Search was utilized for the models with broader parameter options, as testing all parameters would be too time-consuming. Random Search randomly explores only some of the combinations of parameters but can yield similar results to Grid search. The hyperparameter tuning models from scikit-learn could not be used as they require cross-validation, which was not used in this work due to the large dataset. The hyperparameters models were retrieved from Optuna. Optuna is an automatic hyperparameter optimization framework that does not require cross-validation, but only the training and validation sets. It is easy to implement and offers a variety of plots that can provide valuable insights for the

Table 1
Hyperparameters Tuned for Optimal Performance: The table lists the values of the selected hyperparameters, tuned during model optimization to achieve the best performance. The parameter values were initially sampled from widely and logarithmically spaced ranges. For instance, the contamination parameter spanned from 0.05 to 0.3, with a step size of 0.05. However, during the tuning the focus was moved on consist smaller values, closer to the ones that showed the better performance to find the optimal performance of each model.

Models	Parameters	Values
Random Forest with random over-sample	max_iterations	[5,10,15,20,25,30,35]
	max_depth	500
	n_features	None
Isolation Forest	contamination	[0.05,0.1,0.12,0.13,0.14,0.15,0.16,0.17,0.2,0.25,0.3]
	max_samples	[0.3,0.4,0.5,0.6,0.7]
	max_iterations	500
Elliptic Envelope	contamination	[0.05,0.1,0.12,0.13,0.14,0.15,0.16,0.17,0.2,0.25,0.3]
	support_fraction	[0.4,0.5,0.6,0.7,0.8]
	assume_centered	[True, False]
Kernel approximator Nystroem	Kernel	['poly']
	gamma	[0.3,0.4,0.5,0.6,0.7]
	degree	[2,3,4]
One-Class SVM with Stochastic Gradient Descent	coef0	[0.001,0.01,0.1]
	nu	[0.05,0.1,0.15,0.17,0.19,0.2,0.21,0.25,0.3]
	eta0	[0.5,0.6,0.7,0.8,0.9]
	learning_rate	['constant']

hyperparameter tuning, (Akiba et al. 2019). The training set was used to train the model and the evaluation set for the hyperparameter tuning, while the final results were based on the test set.

For the supervised approach, an additional pre-processing step took place, which involved random over-sampling of the minority class. Random over-sampling was used because it ensures that no information from the minority class is lost, as it replicates existing instances and it is easy to implement. Therefore, the RF classifier was tuned twice. The first attempt was made without oversampling while the next one was done with it, making the number of classes equal. An additional advantage of RF is that it is easy to tune. The model was tuned with Grid search, and as seen in Table 1, max_depth was the only parameter tuned, which controls the depth of each decision tree in the forest. Max depth is crucial because if it is set too low the model does not perform well, while if it is too high, it can lead to overfitting. For every hyperparameter trial, the iterations were set to 500, and all the features were included.

Additionally, feature importance was conducted to gain insights into the features, but it was not used for feature selection to ensure that the same features were used in both the supervised and unsupervised approaches for a more accurate comparison between them. After the feature importance, one feature (Bwd Packet Length Std) seemed to have great importance concerning other features as seen in Figure 1. Each network flow consists of more than one packet, and this feature represents the standard deviation of all the packets' length of a network flow when data is transmitted from the destination to the source, and it is crucial because a large value may reveal attempts to disrupt the network by overwhelming them with traffic or attempts to steal data.

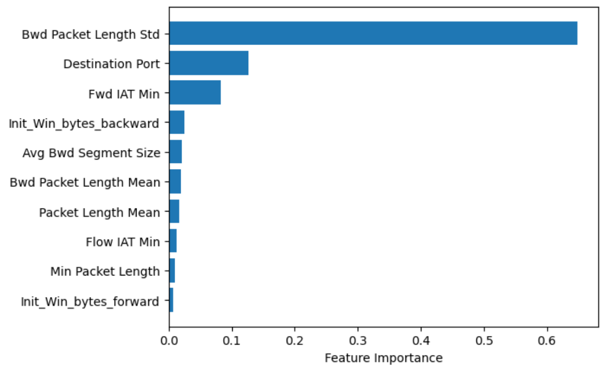


Figure 1
Comparison of Feature Importance in the Random Forest: Notably, the feature ’Bwd Packet Length Std’ demonstrates superior performance, surpassing all other features.

To check if this feature biases or overfits the models’ performance, the hyperparameter tuning of the models was applied again, but it did not have a significant difference in the performance of the models, so it was not excluded. The reason was that other features were highly correlated with it, while others were similar with it, such as "Bwd Packet Length Std" and "Packet Length Std", which combines the standard deviation of the length of the packages for an instance for both the backward and forward direction.

For the unsupervised approach, Grid Search was used for the IF and EE algorithm, as they do not have many parameters to tune. The values of the parameters are illustrated in Table 1. The contamination parameter for both models was tuned, which is the proportion of anomalies that are expected in the dataset, and it is essential because it helps to determine a threshold to identify which instances are an anomaly or not. In IF the iteration was also set to 500, while the max_sample parameter was tuned, where a high value can help in improving slightly the model’s performance, but increase the training time and memory usage, and a lower value may reduce the accuracy but reduce the time and memory needed. For EE, support_fraction and assume_centered were tuned to control the trade-off between the sensitivity of outlier detection and the computational efficiency, and they demonstrated a significant impact on the performance of the model. The OC-SVM with Stochastic Gradient Descend was tuned with the Nystroem kernel approximation, making it challenging to find good parameter values. For that reason, since there were many parameters to tune, and the model was training fast, more trials were made to cover a wider variety of parameters, and Random Search was used. The nu parameter of the OCSVM-SGD-Nystroem is analogous to the contamination parameter of the other two unsupervised models, and eta0 represents the step size of the learning rate, determining the magnitude by which the model parameters are updated based on the computed gradient. The learning_rate, which controls the updating strategy for the eta0, was set to constant, meaning the same eta0 value is used in each step and was the value that improved the model the most. In addition, the Nystroem kernel approximation was also tried with different kernel parameters such as sigmoid, or radial basis function, but the best performance was achieved with the polynomial. The polynomial kernel formula in Equation 3 which was used for the approximation, can help to clarify the parameters that were tuned for Nystroem.

$$K(X,Y) = (\gamma\langle X, Y\rangle + \text{coef0})^{degree} \tag{3}$$

Where X is the vector array that is projected in higher dimensions, and Y is a second optimal array. For more details about the hyperparameter tuning and the parameters' importance see Appendix 2.

Finally, four different metrics were used for the evaluation of the models, namely Precision, Recall, f1-score, and Receiver Operating Characteristic-Area Under the Curve (ROC-AUC). Precision is the ratio of true positive cases to all predicted positive cases, recall is the ratio of true positive cases to all actual positive cases, and f1-score is the harmonic mean of these two. ROC-AUC was used for graphical representation and to summarize the model's overall performance. These metrics are commonly used for intrusion detection, (Sharafaldin, Lashkari, and Ghorbani 2018; Chen et al. 2021). The f1-score was chosen as the primary metric to answer the research questions due to its ability to provide a reliable and comprehensive evaluation of the model's performance since it is robust to class imbalance and the task is binary. An overview of the full data processing and pipeline can be found in Figure 2 and the code at: `https://github.com/Nikolas-Mitsas/Thesis-2023-Intrusion-detection.git`.

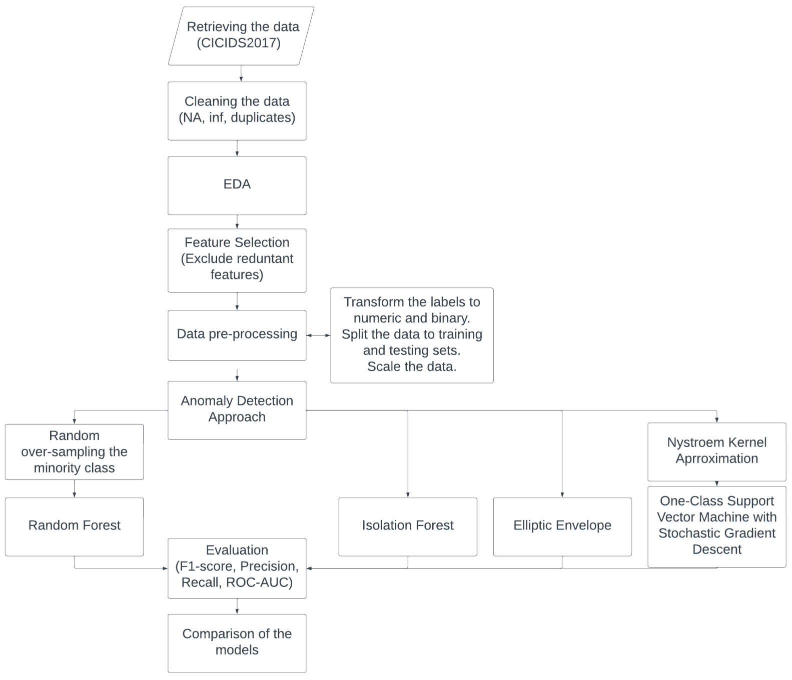

Retrieving the data (CICIDS2017)
Cleaning the data (NA, inf, duplicates)
EDA
Feature Selection (Exclude reduntant features)
Data pre-processing
Transform the labels to numeric and binary. Split the data to training and testing sets. Scale the data.
Anomaly Detection Approach
Random over-sampling the minority class
Random Forest
Isolation Forest
Elliptic Envelope
Nystroem Kernel Aprroximation
One-Class Support Vector Machine with Stochastic Gradient Descent
Evaluation (F1-score, Precision, Recall, ROC-AUC)
Comparison of the models

Figure 2
End-to-End Data Pipeline: Data Retrieval, Cleaning (Handling Missing Values (NA) and Infinite Values (inf)), Pre-processing, Machine Learning Models, and Evaluation Workflow.

4.5 Infrastructure

Hence to hardware limitations, additional computing resources were required for this research, so Google Colab Pro+ was utilized to provide extra memory and processing power. The programming language that was used was Python version 3.10.11 (Van Rossum, Drake et al. 1995). The software libraries employed: Numpy 1.22.4 (Harris et al. 2020), Pandas 1.5.3 (McKinney and Team 2015), Scikit-learn 1.2.2 (Pedregosa et al. 2011), Matplotlib 3.7.1 (Hunter 2007), Optuna 3.1.1 (Akiba et al. 2019), and Seaborn 0.12.2 (Waskom 2021).

5. Results

This section presents the results for both the supervised and unsupervised approaches, on the test set. For the supervised method, the results and the optimal parameters of the RF model are provided. Additionally, the results for the unsupervised models are presented, and their performances are compared with each other. Additionally, we conduct a comparative analysis between the supervised and unsupervised approaches. The evaluation of both approaches is based primarily on the f1-score, and ROC-AUC will also be discussed. Given the potential damages caused by undetected intrusions, accurately identifying true intrusions is the primary objective.

5.1 Topline Performance

Table 2 shows the performance and the best parameters of the RF algorithm. After the hyperparameter tuning of the RF both with and without the use of random over-sampling of the minority class, RF with random over-sampling achieved an f1-score of 99.8%. Those results were slightly higher than the results of RF without the random over-sampling, indicating that RF can still learn to capture the patterns within the classes effectively, regardless of the class imbalance. This can be attributed to the large dataset size, where the minority class still consists of a substantial number of instances. Even though the minority class appears less frequently, its representation is still significant. In addition, there are plenty of features that exhibit high correlations with the classes, which provide valuable insights into the RF model, aiding in the separation of the classes. RF algorithm demonstrates its capabilities to exploit these informative features and make accurate predictions despite class imbalance.

Table 2
Comparison of the best performance in the test set and the optimal parameters. Precision, recall, F1-score, and ROC-AUC values in percentages are reported, for the Random Forest with over-sampling of the minority class and the three unsupervised models: Isolation Forest, Elliptic Envelope, and OCSVM-SGD-Nystroem. Models were trained using 80% of the dataset for training, 10% for hyperparameter tuning, and 10% for testing.

			Evaluation Metrics			
Approach	Models	Parameters	Precision (%)	Recall (%)	F1-score (%)	ROC-AUC (%)
Topline/Supervised Approach	Random Forest with random over-sample	max_iterations=500 max_depth=25 n_features=None	**99.9**	**99.8**	**99.8**	**99.9**
Unsupervised Approaches	Isolation Forest	contamination=0.15 max_samples=0.3 max_iterations=500	55.7	78.1	65.0	85.3
	Elliptic Envelope	contamination=0.13 support_fruction=0.6 assume_centered=True	67.2	**82.5**	74.0	**88.8**
	Kernel approximator Nystroem One-Class SVM with Stochastic Gradient Descent	Kenrel='polynomial' gamma=0.5 degree=3 coef0=0.1 nu=0.2 eta0=0.8 learning_rate=constant	**88.5**	74.2	**80.7**	86.5

In addition, the high value of ROC-AUC, 99.9%, suggests that the model distinguishes the two classes with high precision, as can also be seen from the confusion matrix in Figure 3. RF correctly classified both classes.

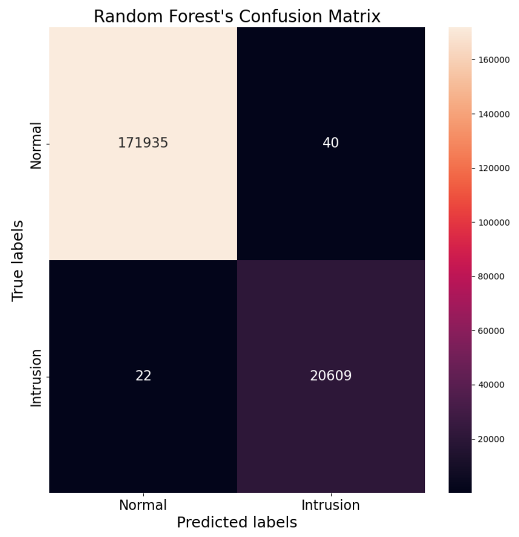


Figure 3
The confusion matrix results for the Random Forest algorithm on the test set. It visualizes the performance of the model and summarizes the actual (y-axis) and predicted (x-axis) values of the model to identify how many were misclassified, shedding light on the efficacy and accuracy of the model.

5.2 Unsupervised approach

The optimal performance of the unsupervised models and the corresponding parameter settings are presented in Table 2. The worst accuracy in the f1-score, 65.0% occurred in IF with a relatively big difference from the other two models, while the ROC-AUC was approximately 85.3%, suggesting the model manage to discriminate the two classes efficiently. The ROC-AUC score was just slightly different from the other two models. Furthermore, the precision is the lowest occurred among the unsupervised models, around 55.7% meaning that half of the alarms from this model will not be intrusions. However, the Recall of IF was higher than that of OCSVM-SGD-Nystroem. Those results indicate that IF misclassified fewer intrusions than OCSVM-SGD-Nystroem but had several more false alarms than any other model.

This is also illustrated in Figure 4, which visualizes and summarizes the IF classification results.

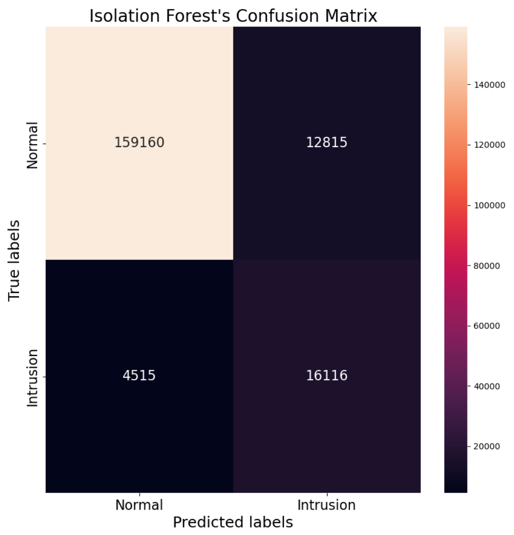


Figure 4
The confusion matrix results for the Isolation Forest algorithm on the test set. It visualizes the performance of the model and summarizes the actual (y-axis) and predicted (x-axis) values of the model to identify how many were misclassified, shedding light on the efficacy and accuracy of the model.

The EE algorithm appears to distinguish the classes better than the other two models and detected the most intrusions, with better precision than IF. However, the precision remains low, indicating many false alarms, as seen in Figure 5. This trade-off between better intrusion detection and increased false alarms should be taken into consideration when evaluating the performance of the EE algorithm. The f1-score of this algorithm was approximately 9% higher than IF, reaching 74.0% which makes it a better choice. However, it seems to have a nearly 6% lower f1-score than OCSVM-SGD-Nystroem. Moreover, based on the ROC-AUC score, the EE algorithm achieved the highest value, of 88.8%, showing that it can distinguish the intrusions from the benign points more accurately than the other two unsupervised models.

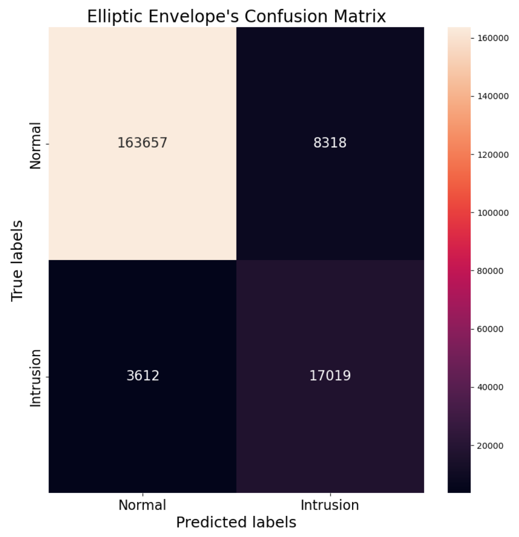


Figure 5
The confusion matrix results for the Elliptic Envelope algorithm on the test set. It visualizes the performance of the model and summarizes the actual (y-axis) and predicted (x-axis) values of the model to identify how many were misclassified, shedding light on the efficacy and accuracy of the model.

Finally, OCSVM-SGD-Nystroem seems to achieve the highest accuracy in the f1-score with 80.7%, while distinguishing the classes efficiently with a ROC-AUC of 86.5%, slightly lower than the EE score. However, the highest f1-score of this model is a result of the high precision that it yields since the recall is the worst one that occurred. In other words, it can identify a significant number of benign points and reduce false alarms, but it does not perform so well in detecting intrusions. Figure 6 and recall, which is 74.2%, suggests that a substantial number of intrusions are not detected, and those can cause several problems. The variation of the misclassified intrusions between OCSVM-SGD-Nystroem and IF is relatively small, while the difference in the false alarms is significant, making the former model preferable. However, it is a difficult choice between this model and EE, which depends on the trade-off between false positives and false negatives. On the one hand, OCSVM-SGD-Nystroem managed the highest f1-score and the fewest false alarms, but it detected fewer intrusions, which is the main focus.

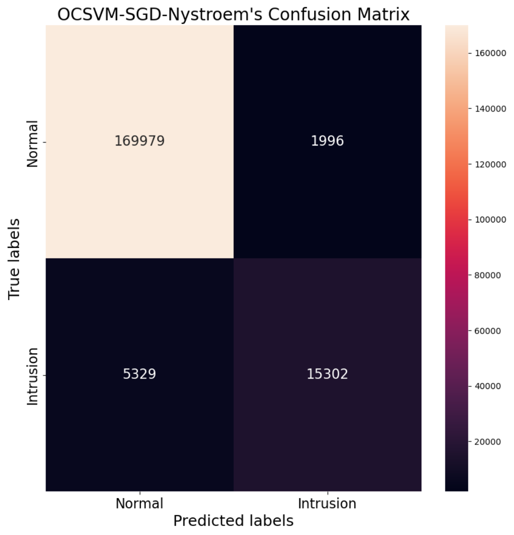


Figure 6
The confusion matrix results for the One-Class SVM with SGD and Nystroem algorithm on the test set. It visualizes the performance of the model and summarizes the actual (y-axis) and predicted (x-axis) values of the model to identify how many were misclassified, shedding light on the efficacy and accuracy of the model.

The variation in false positives and false negatives among the three unsupervised models indicates differences in their ability to detect true intrusions and their sensitivity to anomalies. Some misclassified intrusions appeared the same for all models, indicating a common difficulty in distinguishing those anomalies. While most false alarms are distinct between the models, with a few similar instances revealing a shared blind spot. These findings highlight variations in detection abilities and common challenges faced by the models.

In general, based on the ROC-AUC score of the models, all of them can effectively distinguish and correctly classifies the two classes. The scores were very similar for all the models in comparison to the f1-score that fluctuated among the models, which happens because ROC-AUC focuses more on the instances of the positive class, in this case, normal points, and since there is class imbalance and they are the majority, these results are not reliable. ROC-AUC scores are visualized in Figure 7.

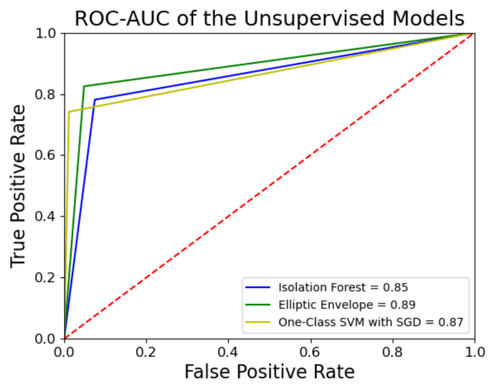


Figure 7
The figure illustrates the performance of each of the unsupervised models in terms of ROC-AUC score. The curve of the model that is closer to the left upper corner is the one with the higher ROC-AUC score and can distinguish the two classes more effectively than the others. The red line in the middle of the plot, the random classifier line, represents the performance of a random classifier that makes predictions purely by chance and has a ROC-AUC score of 0.5.

5.3 Comparison between Supervised and Unsupervised approach

Generally, the supervised approach tends to have much higher performance than the unsupervised, suggesting that the RF model effectively learns from labeled data and captures the underlying patterns within the classes. However, the unsupervised models also demonstrated promising results. More specifically, all the models managed to distinguish the classes effectively, while achieving an overall good accuracy in terms of f1-score, reaching a score of almost 81%. It is important to note that the unsupervised models do not rely on labeled data and can detect intrusions that are considered unknown, making them more suitable for real-world cases where labeled datasets may not be available. This is a significant advantage of unsupervised models compared to supervised models and should be taken into consideration. The choice of the best approach also depends on other factors such as the goal of the IDS that is going to be used.

6. Discussion

6.1 Summary

The focus of this thesis was the creation of IDSs using unsupervised traditional machine learning models. The motivation for this research was that new intrusions tend to appear more often in complex and sophisticated forms, making their detection a challenging task. This was also the reason that unsupervised learning was chosen. As mentioned earlier, most current studies on modern datasets tend to use deep learning models for intrusion detection, whereas previous studies also utilized traditional machine learning algorithms, but these models were trained and evaluated on outdated datasets and although they have shown promising results, they may not generalize well when implemented on current data. Approaches using deep learning models on recent datasets appear to yield high performance, but these models are more complex and time-consuming, requiring additional resources. In this work, traditional machine learning models were used on modern data, as they have shown promising results and require fewer resources.

The dataset used in this work contains updated data. The objective of the thesis was binary, aiming to detect whether a network flow is an intrusion or not, and therefore, the labels were transformed into binary values. First, the data was cleaned and pre-processed to prepare it for implementing the machine learning models. A supervised approach was developed to serve as a topline and was compared to the unsupervised approaches. For the unsupervised approach, three commonly used unsupervised models for anomaly detection were developed and compared to each other and the supervised approach.

6.2 Results

The RF algorithm in the supervised approach yielded significant results, reaching an accuracy of 99.8%. This performance, in the beginning, looked suspicious since it was extremely high. Both precision and recall had similar high results. However, after exploration of the data, it seemed not to have any issues. In addition, comparing it to the supervised approach of Sharafaldin, Lashkari, and Ghorbani (2018), which was in the same dataset and the goal was multi-class intrusion detection, meaning to detect if an instance was benign or which type of the 14 different types of intrusions, the RF algorithm accomplished an accuracy of approximately 98%. In this thesis, where the aim was binary, it was easier for the RF model to detect the intrusions, so it was justified to perform better than that and reach that high accuracy.

For the unsupervised approach, the models in terms of f1-score managed to predict a significant number of intrusions while the number of false alarms among the models fluctuated more. The difference in the abilities of each model in detecting anomalies was clarified from the variation in their performance, mostly based on precision and recall. Additionally, it was shown that the models had different limitations and abilities for detecting intrusions, which may also occur for the different values in the parameters, such as the contamination or nu parameters, that contribute to the decision of the threshold value.

To answer the main research question, the unsupervised traditional machine learning models demonstrated promising results. Comparing this approach with other state-of-the-art, such as in the study of Leung and Leckie (2005), where conventional machine learning models yield similar results. The study evaluated the models' performance using the ROC-AUC metric and outdated data. For example, OC-SVM achieved a ROC-AUC score of around 94%, not so much higher than OCSVM-SGD-Nystroem, that used in this study with an 86.5% ROC-AUC score. In addition, in the study of Belarbi et al. (2022), deep learning models were evaluated in the CICIDS2017 dataset. Comparing the performance of those models, with the ones used in this study demonstrates that traditional machine learning models can yield similar results to more complex approaches. For instants, MLP and LSTM models that were used for binary intrusion detection achieved f1-scores of 87.2% and 89.5%, respectively, not so much higher than the conventional machine learning models utilized in this study. Summarizing these findings, traditional machine learning models can perform similarly to deep learning models for unsupervised intrusion detection and detect intrusions effectively, requiring fewer resources.

Moving to the first research sub-question, the comparison between supervised and unsupervised approaches, the supervised model outperforms significantly each of the unsupervised models, as it was expected. Most of the time, supervised approaches seem to perform better, since in many tasks it is considered less complex. The models are trained with labels, meaning they learn how each class is represented in the data, simplifying the detection process. Despite that, unsupervised models are more complex and tend to result in lower performance, since they are trying to find patterns in the data that will give them insights to make the detection. Moreover, the unsupervised models do not rely on labels and can detect unseen attacks. In conclusion, it is a trade-off between accuracy and reliability since the intrusion is evolving fast, it is best suited for real-world cases to use unsupervised models since they can be more robust to the changes.

Answering the second research sub-question was challenging. IF compared to the other two unsupervised models had less accuracy in terms of f1-score and ROC-AUC, but it manages to detect more intrusions than the OCSVM-SGD-Nystroem. However, its precision indicates that almost half the times the system alerts for intrusions will be wrong, diminishing its functional efficiency and making it the least effective model. The comparison between the EE and OCSVM-SGD-Nystroem was more difficult since the focus of the system was to detect correctly as many intrusions as possible. The EE envelope had the highest accuracy in detecting intrusions but with many false alarms too, dropping its overall accuracy in terms of f1-score. While, OCSVM-SGD-Nystroem yielded the highest overall accuracy of f1-score, and the false alarms were significantly fewer than the other two models. Yet it also found fewer intrusions. Overall it was the best unsupervised model, but since the concern is to avoid probable damages from intrusions EE is more effective in detecting them. More false alarms mean more human intervention to check those alarms and understand whether it was correctly detected. In conclusion, the choice of the best model also depends on the resources at hand.

6.3 Limitation and further research

During this work, the primary limitations that affect the research were the combination of limited resources and the size of the dataset, which made computationally expensive models, impossible to implement. Consequently, models that showed promising results for intrusion detection in previous studies, such as clusters and traditional OC-SVM, were not used. In addition, despite the reduction of the dataset size, the time required for these models remained insufficient for their implementation, but a wider range of parameter values to discover better optimal configurations was achieved.

In future work, it is important to address these limitations and extend the research. Consequently, the complete dataset can be used, so the models can be trained on an extended and more diverse set of instances, potentially leading to higher accuracy and improved generalization. Another avenue to improve the performance of the models could be with a further exploration of the hyperparameter tuning process since we can not be sure that these combinations were the optimal ones, particularly for the OCSVM-SGD-Nystroem that contains many parameters.

For a broader understanding of the efficiency of conventional machine learning models for unsupervised intrusion detection, a variety of frequently used machine learning models can be used, such as clustering algorithms, that have shown promising results in outdated datasets. This also includes evaluating their effectiveness on other datasets to assess their generalizability and potential benefits in intrusion detection.

Additionally, the datasets used for intrusion detection, or anomaly detection, encounter class imbalance since the anomalies are a minority of instances, and handling class imbalance becomes challenging in unsupervised learning. One possible approach is to employ the proposed method of Liu, Ting, and Zhou (2008), Spectral Meta-Learner for Class Imbalance (SMLCI). This ensemble method is for unsupervised models and can also handle class imbalance. Ensemble methods improve the generalization performance of machine learning models, which can significantly improve the overall performance of the models alongside handling class imbalance.

Finally, a comprehensive examination of the outcomes produced by the unsupervised models should be conducted to discern the underlying factors contributing to the notable prevalence of false alarms and the significant divergence in performance when compared to the supervised approach.

7. Conclusion

In conclusion, this study focused on exploring the potential of conventional machine learning models for unsupervised intrusion detection with contemporary data. The results were promising, despite the limited resources, while the choice of the most suited method can vary. The relevance of the supervised approach in addressing real-world problems depends on the users and their intended goals. Although this approach may exhibit higher performance, it may be insufficient in detecting unseen attacks, emphasizing the importance of considering user-specific needs and objectives. The unsupervised approach demonstrated lower but promising results, and as the intrusions are growing fast, it can become more beneficial in future and real-world data. The selection of the best-unsupervised model can be demanding since it also depends on other essential factors and limitations. It is a trade-off between detecting more intrusions with more false alarms or slightly fewer intrusions with significantly fewer false alarms.

This work proposes that traditional machine learning models can be used efficiently for unsupervised intrusion detection in modern network flows and intrusions. Generally, such an approach can be utilized by companies and users that cannot dispose of or do not possess many resources. Computationally it is not expensive, and it can detect a considerably large number of unseen intrusions. In addition, as the disposal of resources may increase, the potential of the models can be increased, achieving an even better accuracy without the need for more complex methods such as deep learning models.

References

Ahmad, Muhammad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. 2021. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using unsw-nb15 data-set. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–23.

Akiba, Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Almuhtaram, Husein, Arash Zamyadi, and Ron Hofmann. 2021. Machine learning for anomaly detection in cyanobacterial fluorescence signals. *Water Research*, 197:117073.

Ashrafuzzaman, Mohammad, Saikat Das, Ananth A Jillepalli, Yacine Chakhchoukh, and Frederick T Sheldon. 2020. Elliptic envelope based detection of stealthy false data injection attacks in smart grid control systems. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1131–1137, IEEE.

Barletta, Vita Santa, Danilo Caivano, Antonella Nannavecchia, and Michele Scalera. 2020. Intrusion detection for in-vehicle communication networks: An unsupervised kohonen som approach. *Future Internet*, 12(7):119.

Belarbi, Othmane, Aftab Khan, Pietro Carnelli, and Theodoros Spyridopoulos. 2022. An intrusion detection system based on deep belief networks. In *Science of Cyber Security: 4th International Conference, SciSec 2022, Matsue, Japan, August 10–12, 2022, Revised Selected Papers*, pages 377–392, Springer.

Breiman, Leo. 2001. Random forests. *Machine learning*, 45:5–32.

Chen, Yang, Nami Ashizawa, Chai Kiat Yeo, Naoto Yanai, and Seanglidet Yean. 2021. Multi-scale self-organizing map assisted deep autoencoding gaussian mixture model for unsupervised intrusion detection. *Knowledge-Based Systems*, 224:107086.

Canadian Institute for Cybersecurity (CIC), University of New Brunswick. 2018. Intrusion detection evaluation dataset (cic-ids2017). https://www.unb.ca/cic/datasets/ids-2017.html. Accessed: 2023-02-24.

Data, Mahendra and Masayoshi Aritsugi. 2021. T-dfnn: An incremental learning algorithm for intrusion detection systems. *IEEE Access*, 9:154156–154171.

Gharib, Amirhossein, Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2016. An evaluation framework for intrusion detection dataset. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–6, IEEE.

Harris, Charles R, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with numpy. *Nature*, 585(7825):357–362.

Hunter, John D. 2007. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95.

Kemmerer, Richard A and Giovanni Vigna. 2002. Intrusion detection: a brief history and overview. *Computer*, 35(4):supl27–supl30.

Lanvin, Maxime, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, Ludovic Mé, and Eric Totel. 2022. Errors in the cicids2017 dataset and the significant differences in detection performances it makes. In *International Conference on Risks and Security of Internet and Systems*, pages 18–33, Springer.

Lee, Wenke, Salvatore J Stolfo, and Kui W Mok. 1999. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*, pages 120–132, IEEE.

Leung, Kingsly and Christopher Leckie. 2005. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422, IEEE.

McKinney, Wes and PD Team. 2015. Pandas-powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit*, 1625.

Modirrousta, Mohammad Hossein, Parisa Forghani, and Mahdi Aliyari Shoorehdeli. 2022. Analysis of anomalous behavior in network systems using deep reinforcement learning with cnn architecture. *arXiv preprint arXiv:2211.16304*.

Panigrahi, Ranjit and Samarjeet Borah. 2018. A detailed analysis of cicids2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology*, 7(3.24):479–482.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Poon, Lex, Siamak Farshidi, Na Li, and Zhiming Zhao. 2021. Unsupervised anomaly detection in data quality control. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2327–2336, IEEE.

Qu, Xiaofei, Lin Yang, Kai Guo, Linru Ma, Meng Sun, Mingxing Ke, and Mu Li. 2021. A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile networks and applications*, 26:808–829.

Ravipati, Rama Devi and Munther Abualkibash. 2019. Intrusion detection system classification using different machine learning algorithms on kdd-99 and nsl-kdd datasets-a review paper. *International Journal of Computer Science & Information Technology (IJCSIT) Vol*, 11.

Rousseeuw, Peter J and Katrien Van Driessen. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223.

Schölkopf, Bernhard, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support vector method for novelty detection. *Advances in neural information processing systems*, 12.

Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116.

Sinclair, Chris, Lyn Pierce, and Sara Matzner. 1999. An application of machine learning to network intrusion detection. In *Proceedings 15th annual computer security applications conference (ACSAC'99)*, pages 371–377, IEEE.

Van Rossum, Guido, Fred L Drake, et al. 1995. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Waskom, Michael L. 2021. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021.

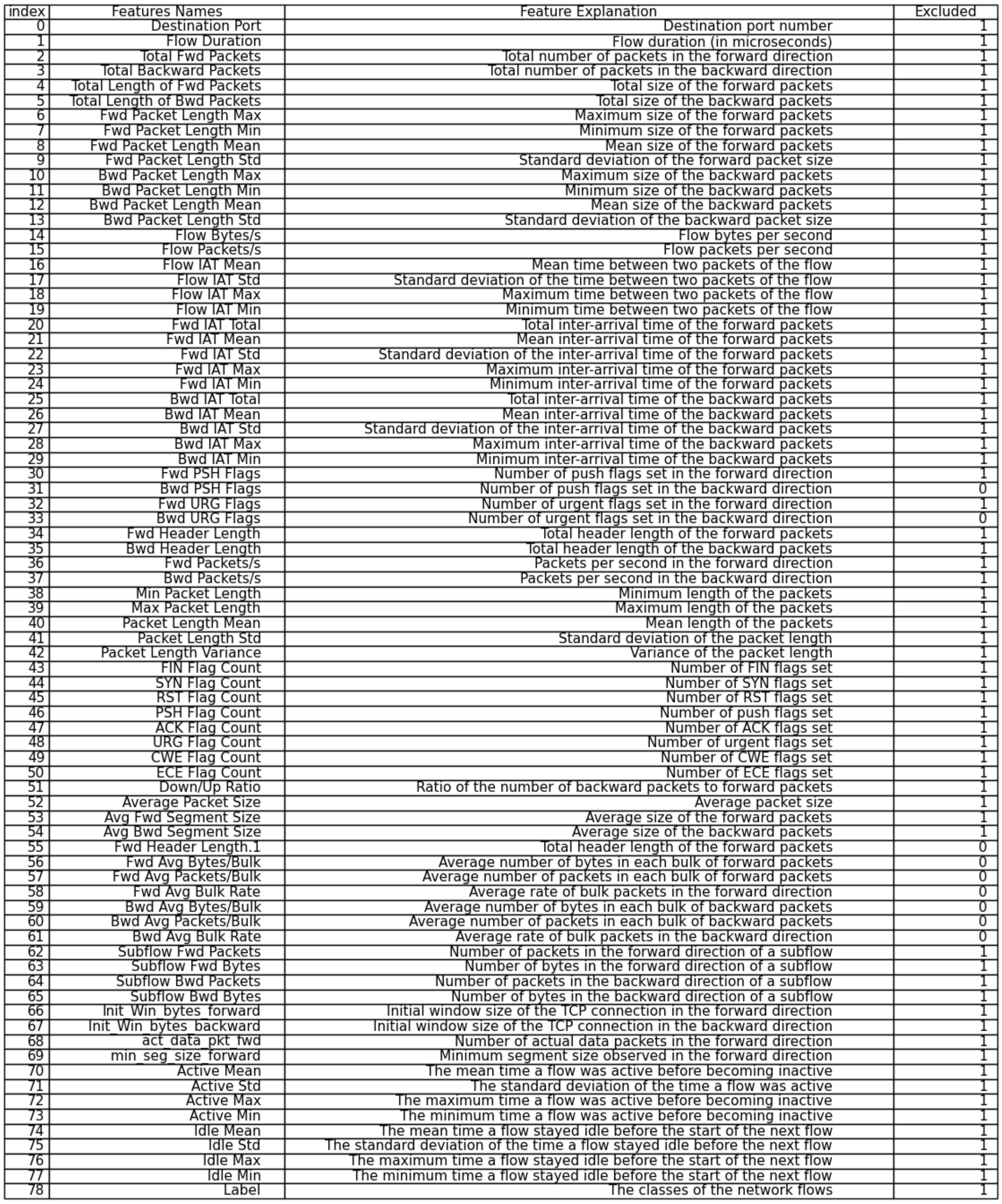
Williams, Christopher and Matthias Seeger. 2000. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13.

Yang, Tianbao, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. 2012. Nyström method vs random fourier features: A theoretical and empirical comparison. *Advances in neural information processing systems*, 25.

Zhang, Jiong and Mohammad Zulkernine. 2006. Anomaly based network intrusion detection with unsupervised outlier detection. In *2006 IEEE International Conference on Communications*, volume 5, pages 2388–2393, IEEE.

Zhang, Zhao, Yong Zhang, Da Guo, and Mei Song. 2021. A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks. *International Journal of Machine Learning and Cybernetics*, 12:1649–1665.

1. Appendix



index	Features Names	Feature Explanation	Excluded
0	Destination Port	Destination port number	1
1	Flow Duration	Flow duration (in microseconds)	1
2	Total Fwd Packets	Total number of packets in the forward direction	1
3	Total Backward Packets	Total number of packets in the backward direction	1
4	Total Length of Fwd Packets	Total size of the forward packets	1
5	Total Length of Bwd Packets	Total size of the backward packets	1
6	Fwd Packet Length Max	Maximum size of the forward packets	1
7	Fwd Packet Length Min	Minimum size of the forward packets	1
8	Fwd Packet Length Mean	Mean size of the forward packets	1
9	Fwd Packet Length Std	Standard deviation of the forward packet size	1
10	Bwd Packet Length Max	Maximum size of the backward packets	1
11	Bwd Packet Length Min	Minimum size of the backward packets	1
12	Bwd Packet Length Mean	Mean size of the backward packets	1
13	Bwd Packet Length Std	Standard deviation of the backward packet size	1
14	Flow Bytes/s	Flow bytes per second	1
15	Flow Packets/s	Flow packets per second	1
16	Flow IAT Mean	Mean time between two packets of the flow	1
17	Flow IAT Std	Standard deviation of the time between two packets of the flow	1
18	Flow IAT Max	Maximum time between two packets of the flow	1
19	Flow IAT Min	Minimum time between two packets of the flow	1
20	Fwd IAT Total	Total inter-arrival time of the forward packets	1
21	Fwd IAT Mean	Mean inter-arrival time of the forward packets	1
22	Fwd IAT Std	Standard deviation of the inter-arrival time of the forward packets	1
23	Fwd IAT Max	Maximum inter-arrival time of the forward packets	1
24	Fwd IAT Min	Minimum inter-arrival time of the forward packets	1
25	Bwd IAT Total	Total inter-arrival time of the backward packets	1
26	Bwd IAT Mean	Mean inter-arrival time of the backward packets	1
27	Bwd IAT Std	Standard deviation of the inter-arrival time of the backward packets	1
28	Bwd IAT Max	Maximum inter-arrival time of the backward packets	1
29	Bwd IAT Min	Minimum inter-arrival time of the backward packets	1
30	Fwd PSH Flags	Number of push flags set in the forward direction	1
31	Bwd PSH Flags	Number of push flags set in the backward direction	0
32	Fwd URG Flags	Number of urgent flags set in the forward direction	1
33	Bwd URG Flags	Number of urgent flags set in the backward direction	0
34	Fwd Header Length	Total header length of the forward packets	1
35	Bwd Header Length	Total header length of the backward packets	1
36	Fwd Packets/s	Packets per second in the forward direction	1
37	Bwd Packets/s	Packets per second in the backward direction	1
38	Min Packet Length	Minimum length of the packets	1
39	Max Packet Length	Maximum length of the packets	1
40	Packet Length Mean	Mean length of the packets	1
41	Packet Length Std	Standard deviation of the packet length	1
42	Packet Length Variance	Variance of the packet length	1
43	FIN Flag Count	Number of FIN flags set	1
44	SYN Flag Count	Number of SYN flags set	1
45	RST Flag Count	Number of RST flags set	1
46	PSH Flag Count	Number of push flags set	1
47	ACK Flag Count	Number of ACK flags set	1
48	URG Flag Count	Number of urgent flags set	1
49	CWE Flag Count	Number of CWE flags set	1
50	ECE Flag Count	Number of ECE flags set	1
51	Down/Up Ratio	Ratio of the number of backward packets to forward packets	1
52	Average Packet Size	Average packet size	1
53	Avg Fwd Segment Size	Average size of the forward packets	1
54	Avg Bwd Segment Size	Average size of the backward packets	1
55	Fwd Header Length.1	Total header length of the forward packets	0
56	Fwd Avg Bytes/Bulk	Average number of bytes in each bulk of forward packets	0
57	Fwd Avg Packets/Bulk	Average number of packets in each bulk of forward packets	0
58	Fwd Avg Bulk Rate	Average rate of bulk packets in the forward direction	0
59	Bwd Avg Bytes/Bulk	Average number of bytes in each bulk of backward packets	0
60	Bwd Avg Packets/Bulk	Average number of packets in each bulk of backward packets	0
61	Bwd Avg Bulk Rate	Average rate of bulk packets in the backward direction	0
62	Subflow Fwd Packets	Number of packets in the forward direction of a subflow	1
63	Subflow Fwd Bytes	Number of bytes in the forward direction of a subflow	1
64	Subflow Bwd Packets	Number of packets in the backward direction of a subflow	1
65	Subflow Bwd Bytes	Number of bytes in the backward direction of a subflow	1
66	Init_Win_bytes_forward	Initial window size of the TCP connection in the forward direction	1
67	Init_Win_bytes_backward	Initial window size of the TCP connection in the backward direction	1
68	act_data_pkt_fwd	Number of actual data packets in the forward direction	1
69	min_seg_size_forward	Minimum segment size observed in the forward direction	1
70	Active Mean	The mean time a flow was active before becoming inactive	1
71	Active Std	The standard deviation of the time a flow was active	1
72	Active Max	The maximum time a flow was active before becoming inactive	1
73	Active Min	The minimum time a flow was active before becoming inactive	1
74	Idle Mean	The mean time a flow stayed idle before the start of the next flow	1
75	Idle Std	The standard deviation of the time a flow stayed idle before the next flow	1
76	Idle Max	The maximum time a flow stayed idle before the start of the next flow	1
77	Idle Min	The minimum time a flow stayed idle before the start of the next flow	1
78	Label	The classes of the network flows	1

Figure 8
The table contains the features' names, the explanations, and the values that were dropped, which are represented with a zero value in the last column.

.

2. Appendix

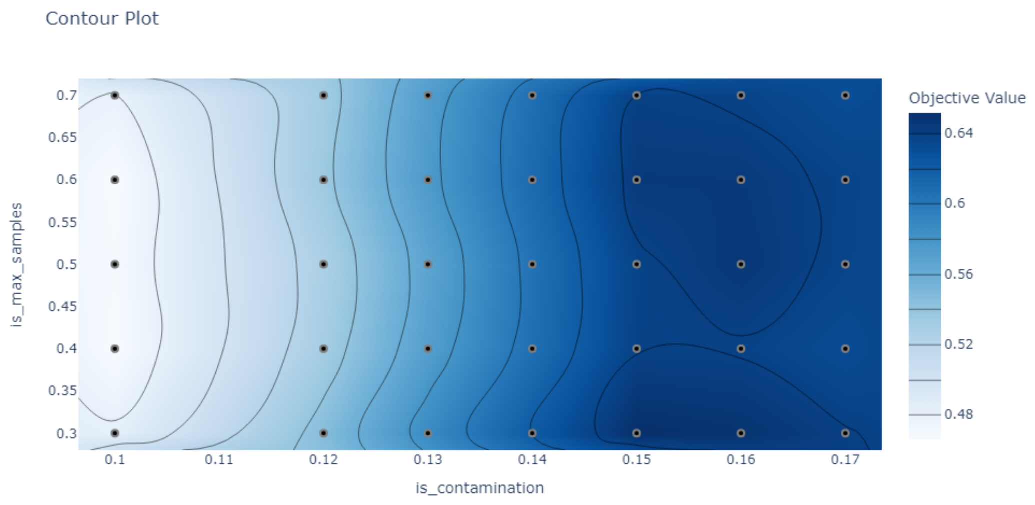


Figure 9
The plot, retrieved from the hyperparameter tuning with Optuna, illustrates the influence of the parameters, contamination, and max_sample, that were tuned for the IF. It reveals that with a lower contamination value, the performance is reduced, while the max_sample does not have a big influence on the performance. For both parameters additional numbers were tested but did not yield better results.

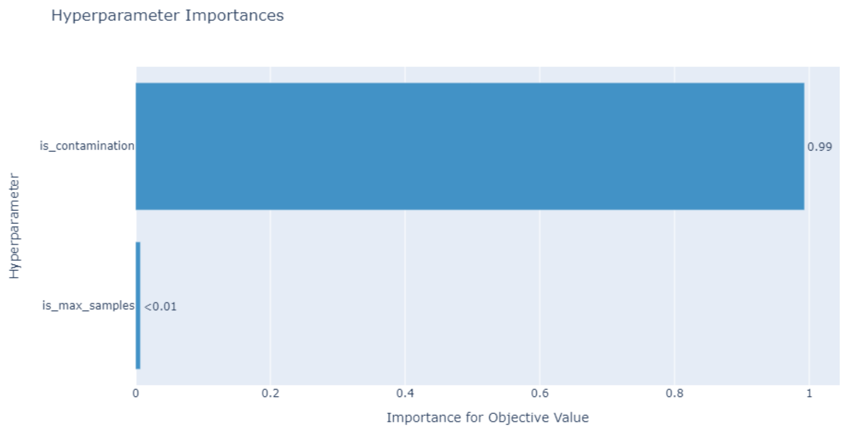


Figure 10
The parameter importance of the two parameters there were tuned for IF. Max_sample had a very small impact, while contamination determined the final performance.

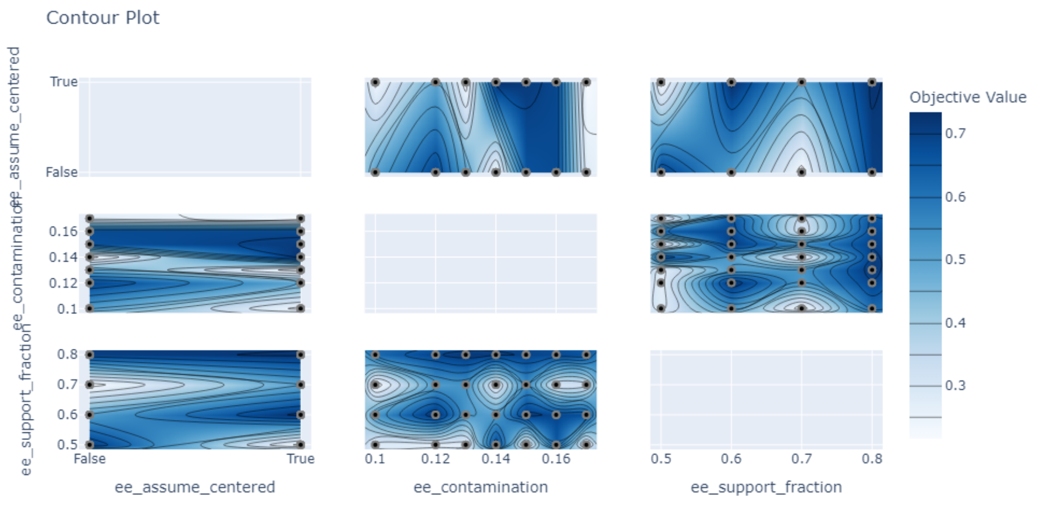


Figure 11
The plot, retrieved from the hyperparameter tuning with Optuna, illustrates the influence of the three parameters, that were tuned for the EE. Higher accuracy seems to occur more often when assume_centered is set to true. The value for the contamination fluctuates a lot and highly depends on the value of the support_fraction. The plot is focused on the values that were close to the optimal performance.

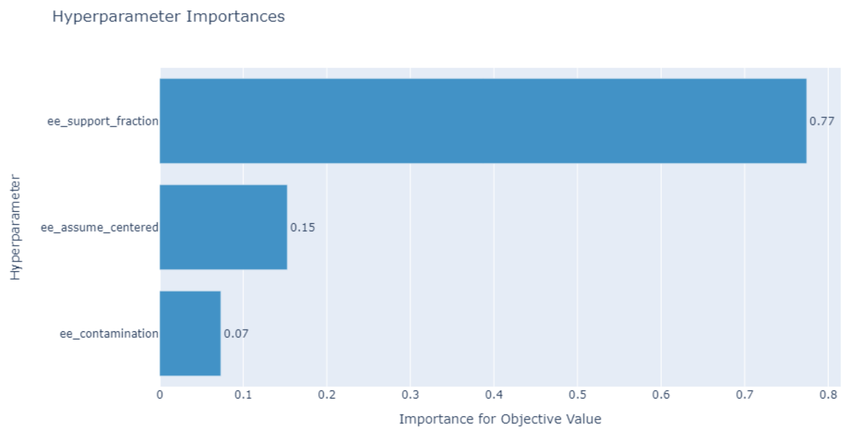


Figure 12
The parameter importance of the three parameters there were tuned for EE. For EE the contamination parameter was the least important, while support_fraction had the highest impact on the results of this model.

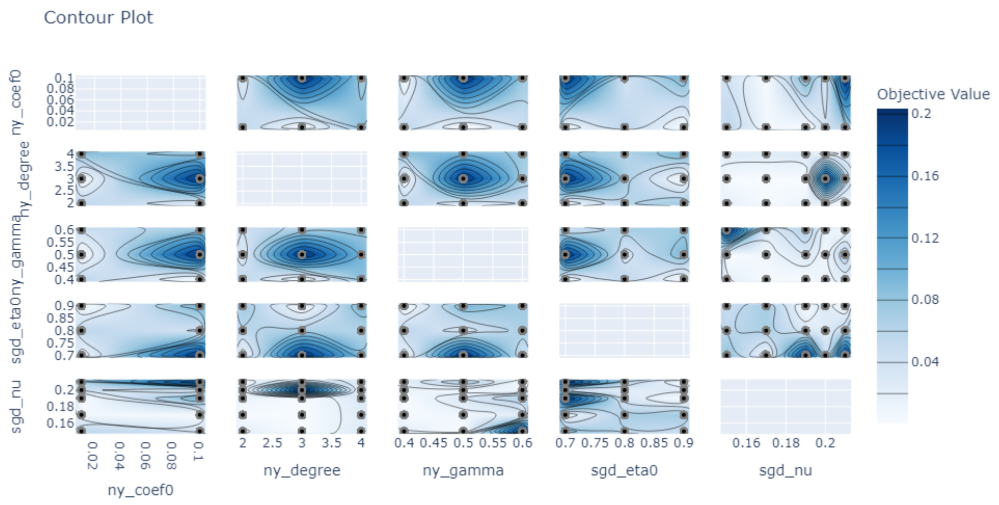


Figure 13
The plot, retrieved from the hyperparameter tuning with Optuna, illustrates the influence of the parameters, that was tuned for the OCSVM-SGD-Nystroem. The combination of parameters for this model was more challenging since the number of parameters and the number of trials were larger than the other models. In all combinations, the degree parameter with value 3 has the highest performance and the coef0 seems to perform better with a value closer to 0.1. In addition, eta0 also performs better with values close to 0.7, while for the nu and gamma parameters, their optimal value depends on the values of the other parameters. The parameters with ny belong to the Nystroem kernel approximation and the ones with sgd to the OC-SVM with Stochastic Gradient Descend. The plot is focused on the values that were close to the optimal performance.

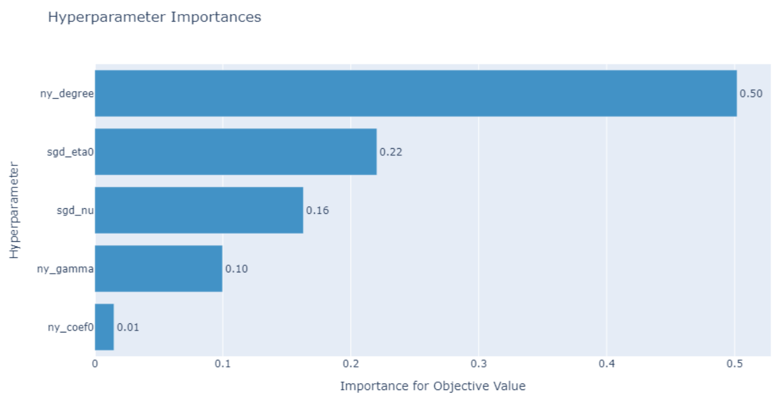


Figure 14
The parameter for the OCSVM-SGD-Nystroem, were tuned together. The figure illustrates the parameter importance of the parameters for both models together. The degree is the most important parameter since it determines the complexity of the polynomial kernel. Then gamma and coef0 are the least important which are also parameters of the Nystroem kernel approximation. For the OCSVM-SGD-Nystroem, nu is the most important parameter, as it helps to separate the anomalies for the normal data, while also eta0 affects the final results. The choice of kernel and the learning_rate of OC-SVM with Stochastic Gradient Descend were also tuned but they are not illustrated, because this was the final model tuned, with the highest accuracy. The parameters with ny belong to the Nystroem kernel approximation and the ones with sgd to the Stochastic Gradient Descend.