SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Data Mining for Cyber Security

Progetto di Ingegneria Informatica
Computer Science Engineering - Ingegneria informatica

Emilio Fattibene - Virgilio Cusano, 10812486 - 10813054

Advisor:
Prof. Maria Fugini

Duration:
From December 2024 to April 2025

Abstract: With the recent surge in internet usage, the volume of confidential data online has increased. To prevent attackers to gain access to these private information, several systems are implemented. With the rapid growth of network traffic, malicious users are rapidly developing new methods of network intrusion. To face these threats, the security systems have to develop high accuracy, short response time, and a never seen agility in recognizing the before-never-seen threats. One of the most crucial security systems is certainly the Intrusion Detection System (IDS). This research explores machine learning approaches for IDS. One dataset was utilized in this study: UNSW-NB15. In this article we propose an intrusion detection system based on ensemble voting. We perform testing on real-world data using an unbalanced database using four different classification algorithms: Decision Tree (DT), Random Forest (RF), K-nearest neighbor (KNN), and Multiple Layer Perceptron (MLP). Furthermore, the voting ensemble classification method is used to improve the accuracy of the model and reduce the number of false-positives. By using the deep learning we also increase the possibility of discovery of new attacks. This research also has a goal to increase the explainability of anomaly-based NIDS, a problem largely ignored in literature.

Key-words: IDS, Deep learning, Machine learning, Ensemble learning, UNSW-NB15, XAI

1. Introduction

In an increasingly digital and diverse environment, it is extremely difficult to protect private information and digital infrastructures using only traditional security solutions. From recent reports, it is clear that there is an increasingly number of the most common types of Cybersecurity Attacks, such as: DDos, MITM attacks, Phishing and Ransomware. In fact, in 2023 the number of DDos attacks per user has grown by 94% compared to 2022 (see [15]), which makes clear that there is a huge lack of defenses against these attacks.

In the The $(CS)^2$ AI-KPMG Control System Cybersecurity Annual Report [4], KPMG classified IDSs as the most used security technology to protect organization control system assets against cyber threats.

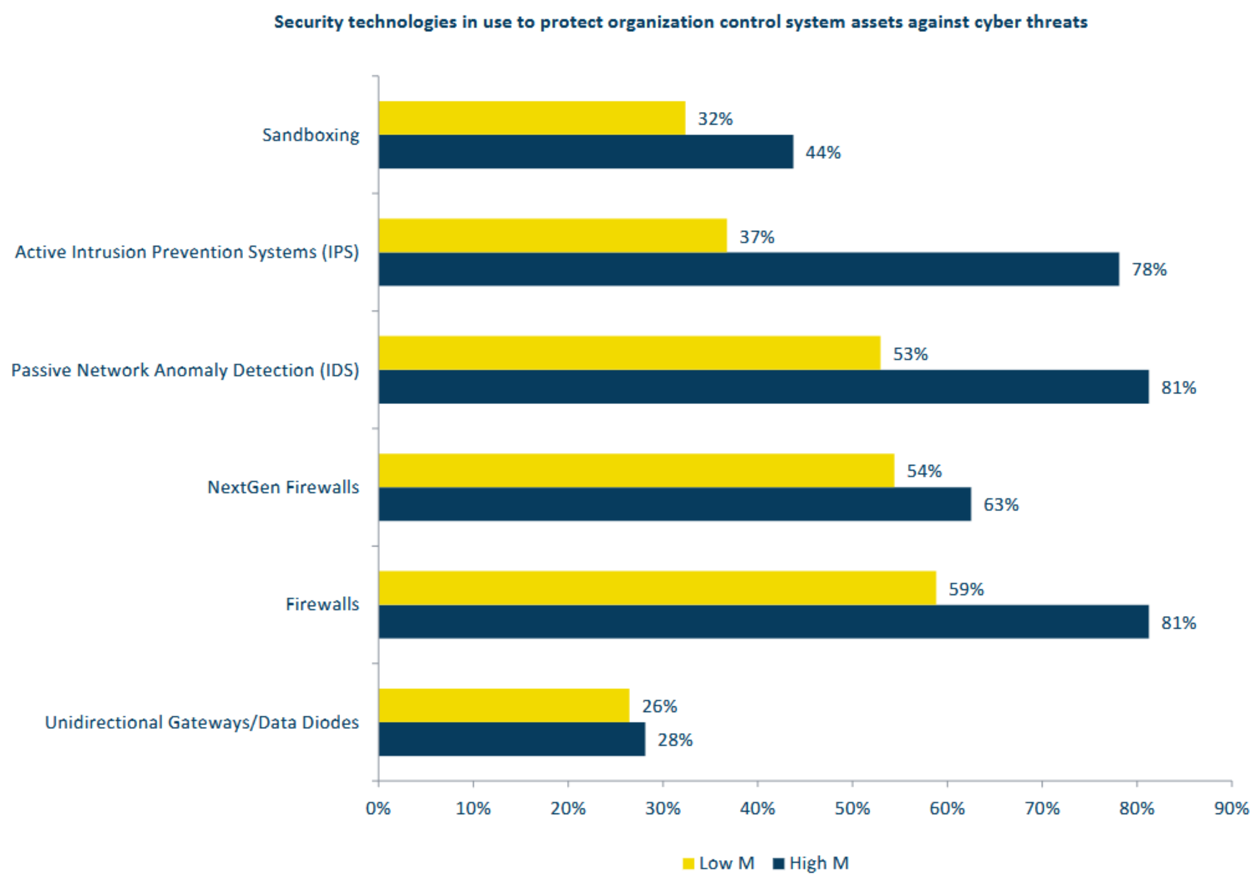


Figure 1: $(CS)^2$ AI-KPMG 2024 report

A IDS (Intrusion Detection System) is a mechanism that monitors network traffic to detect suspicious or abnormal activity and then takes actions based on its type. IDSs have several classifications depending on different variables. The most common ones are based on information source, type of analysis and type of response.
Information Source:

- Network-based (NIDS): The IDS analyze the network traffic (es. packets) to detect possible attacks or intrusion. It requires distributed sensors in various strategic points of the network. The pros are that it needs few sensors to monitor even a big network but the cons are the low efficiency in a network congestion.
- Host-based (HIDS): The IDS is installed on every machine that needs to be protected and monitors process and application on the host where is installed. It has an high precision, especially in detecting Trojan Horse and Worms but has also a complex management system due to his poor scalability.

Type of analysis:

- Signature detection: This analysis is meant to detect pattern called *signature* of behavior related to known attacks. This type of IDS has a low rate of False Positive and are characterized with a fast reaction. The main downside of using a signature-based IDS is the inability of these systems to recognize unknown attacks.
- Anomaly detection: On the opposite side compared to signature detection, this analysis focuses on anomaly action without confronting it with previous pattern attack. This IDS detect significative deviation from the normal behavior of a system confronting them with model build from history data. Although the high percentage of False Positive, these IDS can detect unknown

attacks.

Type of response:

- Active Response: The IDS can decide the course of action. An active IDS and take action by resetting the connection or by reconfiguring the firewall trying to block the underway attack.
- Passive Response: The IDS notifies the system administrator reporting the attack and delegates to specialized operators the solution to prevent damage.

The Intrusion Detection System proposed in this article is a passive anomaly-based NIDS. While IDSs in general require high accuracy and a short response time, passive IDSs, which we said provide results meant to be seen by an operator, require additional care on transparency and report readability. The proposed solution produces a report, both in csv and xlsx format, meant to be easy to look at. The goal of an easy-to-read report is that an expert operator can easily determine whether the IDS result is correct and implement the best policy to combat the signaled threat, reducing the comprehensive response time. To achieve such feat we selected a minimal complete partition of the collected features. Since the volume of network flow is in a continuing growth, not only machine learning becomes fundamental for cybersecurity, but it requires additional care. Commonly used classification algorithms, such as Decision Tree and Random Forest provide high accuracy but they can be prone to problems such as overfitting. To anticipate and solve these problems feature selection was used. The main advantages of using feature selection are:

- Avoiding the curse of dimensionality;
- Avoiding overfitting;
- Reducing the comprehensive time required for the model fitting and prediction;
- Facilitating the model interpretability.

Feature selection can help increasing precision and lowering recall, but the final results strongly depends on the classification algorithm's precision. To exceed the classifier's limits an ensemble method is provided. Using ensemble methods, in this case Voting, can reduce the false positives and increase accuracy and prediction. This improvement is mostly compensated by the time required for the ensemble method. To reduce time consumption, parallelization and dataset reduction techniques are used.

2. Related work

In recent years, IDS research has mainly focused on improving accuracy and adaptability to new threats through the implementation of machine learning and deep learning techniques. For example, through the adoption of deep learning it has been possible to achieve high accuracy in the binary classification of DDoS attacks (see [1]). Given its increasingly frequent use, IDS have been used in various fields, such as the automotive one, with the aim of monitoring the Controller Area Network (CAN) of vehicles through the adoption of hybrid neural networks (see [13]). In the case of this research, the project aims to provide an adaptive and precise solution, with the possibility of use in various fields, providing as an innovation the use of ensemble voting with the aim of increasing the accuracy of the results. In fact, not only were all best practices followed and the most performing algorithms in current literature used, but it was done in such a way as to guarantee a low rate of false positives and false negatives through particular attention to voting management.

One of the critical issues that must be addressed in network security is implementing an effective network intrusion detection. Despite the progresses made in this field, the majority of Network-IDS are signature-based, leading a complex network exposed to zero-day attacks and alternative version of known attacks. In particular, an increasing number of systems are adopting signature-based IDS rather than anomaly detection IDS for reasons like durability of training data, high costs and error rates owing to the dynamic nature of the data. Progressively, IDSs are evolving as market product letting companies use them as COTS, as they are becoming the primary subject and direction of Internet security research. Technology for intrusion detection is in constant develop thanks to the incorporation of technologies like: *data mining*, that allow an IDS to perform a more efficient and accurate training, *pattern recognition*, enhancing signature-based systems, and *neural, networks* that has improved the

speed in the recognition phase.
Talking about adaptability to new and unforeseen attack patterns, recent studies done by Yoheswari (see [12]) shows that by updating periodically the dataframe used to train the model with new and before unknown attacks, could significantly increase the accuracy, false positive rate and computational efficiency. As a matter of fact, future work could possibly implement a mechanism to update the data base used by the model with new behavior discovered during activity recognition, enhancing the accuracy of future prediction.
Dini et al, [2] produced an interesting overview on Intrusion Detection Systems, comparing different classifiers performances on three different datasets. This is one of the few studies that compare not only binary classification and multi-class classification, but also compares computation time. It is noted that multi-class classification is lacks in precision and is more computationally demanding compared to binary classification.
Hnamte & Hussain [3] propose an hybrid deep learning-based IDS. The proposed model is heavily focused on deep learning and achieved a 99.64% accuracy when tested. While the performance is outstanding, the computational time is not discussed.
Milosevic, Ciric, and Milentijevic [9] explore a weighted voting ensemble method that uses deep learning-based classifiers. This model, tested using the CICIDS-2017 dataset, is unable to identify Web Attacks but produces otustading results in all the other categories. Again, in the paper is not explored computational time, which could be problematic if using a deep learning-based ensemble method.
Sridevi et al. [14] propose a supervised learning based voting classifier. The proposed model produces optimal results, but the chosen dataset was collected more than 25 years ago and it contains only four categories of attacks. Deep learning-based classifiers are also not explored as well as computational time.
Jain et al. [5] propose an IDS that uses an ensemble technique composed of eleven models. The proposed classifiers are both machine learning-based and deep learning-based. Memory-managing classifiers such as LightGBM are also explored, providing an insight on future developments.
In a nutshell, deep learning-based IDSs and voting-based IDSs are both explored in the literature and provide promising results. On the other hand, their computational cost is seldom explored and discussed, even though is extremely relevant in NIDS.

3. Innovations

The proposed model combines several good-practices and recently proposed structures to achieve high accuracy, low false-positives rate and high explainability. The significant innovation of our research resides in the analysis of time complexity and explainability, both crucial for Intrusion Detection Systems and ignored in literature. We developed a state-of-the-art model that combined literature's newest innovations and this newly implemented analysis.

3.1. Data preprocessing

Datasets usually contain features which may affect the performance of some machine learning algorithms. Gradient descent based algorithms, like neural networks, are greatly affected by this. Following these principles, we used a several label encoders to transform the values of the string-based columns in a numeric value. The label encoder implemented by sklearn uses a numerical codific. Due to the structure of the problem, additional pre-processing, such as PCA, would have led to an analysis in a eleven-dimensional space. For this reason we didn't produce any additional pre-processing analysis. A scaler was also applied before using the KNN classification algorithm and the MLP classification algorithm. For the KNN, a MinMax scaler was used, while for the MLP was chosen a StandardScaler. The small amount of data preprocessing required is due to the chosen dataset. The UNSW-NB15 dataset, which will be discussed later, provides an already preprocessed dataset. Raw data, which we decided to not use, were also available. To include this model in a more complex system, data preprocessing would be fundamental.

3.2. Ensemble method

Ensemble classification is a result of the Byzantine Generals Problem[7]. Ensemble classification relies on the credibility of the singular components and provides a result based on the convolution of the singular results. Ensemble methods construct a set of several classifiers (in this research we limited the set to 3 classifiers) and provide a result based on a convolution of the results of the classifiers, To provide the result, ensemble methods offers several mechanisms, such as voting, stacking, and bagging. Ensemble classifiers offer the advantage of achieving higher accuracy and lower false-positives rate than the original classifiers. Weighting the classifiers, which is the analog version of the hyperparameter setting for the singular classifier, can help achieving better results.
Ensemble methods can be classified in two distinct groups:

- Parallel: Parallel models train their models independently. This method doesn't drastically increase the time complexity if the mathematical parallelization is followed by a physical parallelization of the classifiers on several cores.

$$T_{Parallel} = \max(\{T_i | i \in [0, ..., n]\}) \tag{1}$$

  with n being the number of classifiers used by the model.
- Sequential: Sequential models use the result of the previous classifier in order to reduce the total error of the ensemble method. These methods can be more time consuming because of their sequential nature.

$$T_{Sequential} = \sum_{i=0}^{n} T_i \tag{2}$$

  with n being the number of classifiers used by the model.

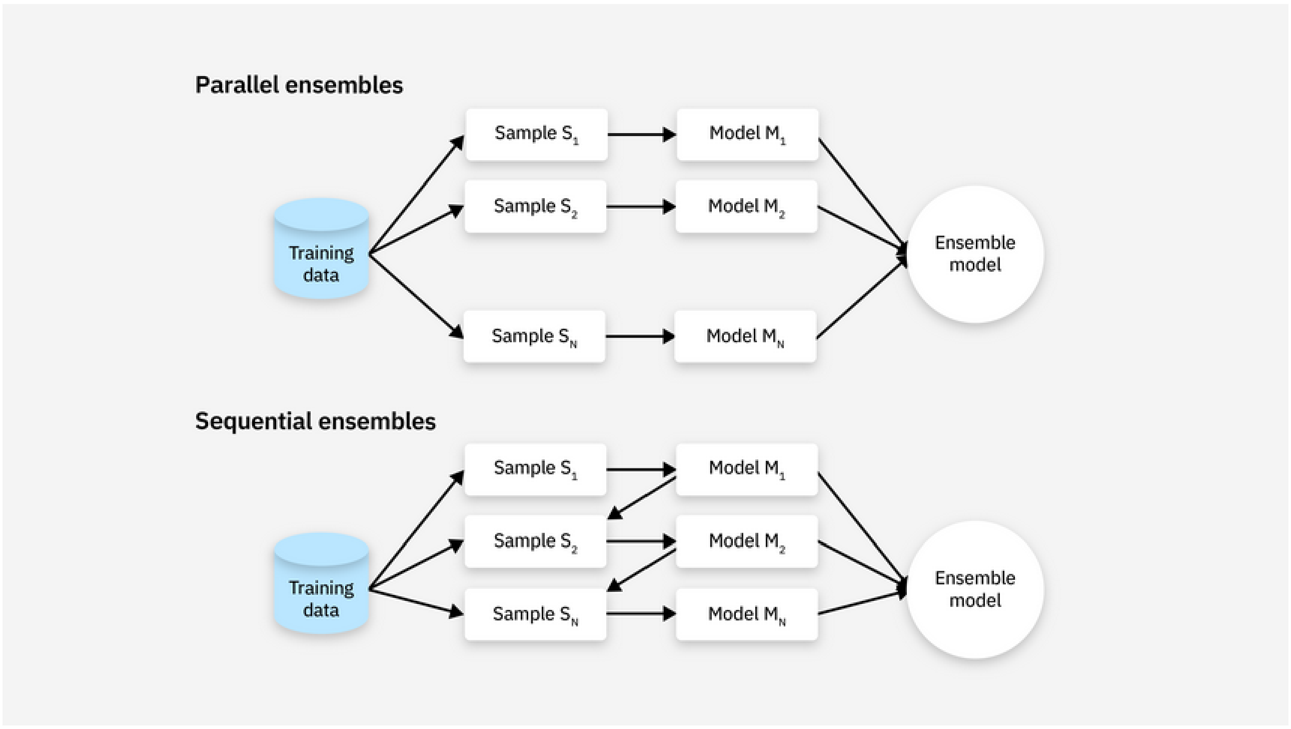


Figure 2: https://www.ibm.com/think/topics/ensemble-learning

In this research, a parallel voting ensemble classifier was used to achieve higher accuracy and to reduce false positives. The voting classifier used four classifiers, which are DT, RF, MPL, and KNN. To reduce time-complexity, parallelization was used, allocating one core to every classifier. The weight distribution was also changed from the standard (1,1,1,1) to a more balanced (3,3,1,2). The weight was based on the trust derived from the individual performances.

3.3. Deep learning

Deep learning refers to a family of algorithms that uses hidden layers to calculate the prediction. We used the Multiple Layer Perceptron classification algorithm (MLP), which uses several hidden layers to predict the correct value. A Perceptron[11], as the original definition states, is a system which learn to recognize similarities or identities between patterns of optical, eletrical, or tonal information, in a manner which may be closely analogous to the perceptual processes of a biological brain. The MLP works by feeding to the i-th layer the results of the previous layer's perceptrons. When the last layer predicts the result, a filter such as the Rectified Linear Unit (ReLU) function or the Adam solver[6] is applied and the result is recalculated until the model achieves a satisfactory result. This system allows the classification algorithm to be more agile and not prone to phenomenon such as overfitting or underfitting. The cons resides in low explainability and in the fact that these systems are susceptible to normalization. Deep Learning for malicious packets detection is a double-edged sword. On the one side it achieves better results than any other model in detecting the least-represented attacks, on the other side it usually drops the precision of the model and reduces the explainability.
Deep Learning algorithms also usually have higher computational complexity than the other classifiers.

3.4. Explainability (XAI)

Explainability is a matter of strong interest in recent researches. Explainability regards the ability of a model to provide an understandable explanation for its predictions. This ability is crucial for building trust and confidence in the model. An organization which uses deep-learning and AI has to keep this in mind.
Explainability is pivotal in the proposed model. Explainability in a passive IDS is crucial because of the role of the human operator that interprets it's results. In particular, explainability for IDSs produces advantages both in reducing comprehensive analysis time, both in incrementing the human operator accuracy itself.
To increase our model explainability, we chose to deliver the IDS's output in a xlsx format, which is easy to interpret, providing only the necessary and most meaningful information about the identified malicious transaction. Because the model is set, assuming the absence of a technical issue that led to the reset of the system, it is easy to request additional information regarding the identified transaction. This request could possibly lead to a written response indicating all the contents of the identified transaction, a written response indicating the decision path of the decision tree and the random forest, or even a picture showing the decision paths. Because this request requires a larger system, this particular feature is not implemented in our solution but our system allows it's possibility. In case of a technical issue, there is no guarantee of this feature's correctness.
The reader has to keep in mind that explainability regards the model itself and does not exceed these limits: the model provides a result that is easy to look at and interpret, and it should be clear why the model predicted a particular instance as such. How the operator decides to operate once the result is provided and which action does he has to take are considerations regarding the operator's policies, not the model itself.

3.5. Time efficiency

Time performances are rarely covered in the literature. The proposed system not only is able to classify with high accuracy, but implements several mechanisms to reduce time complexity.
The classification algorithms are chosen keeping in mind their time complexity. Furthermore, the ensemble model is trained and classifies on several cores, reducing the duration required by both actions via parallelization.

The testing dataset was also divided in batches to simulate a real network flow. The batches dimension was set to 2000, but in a future application may be better to adjust dynamically the size of the batches, scaling its value by considering both the thread pool and the incoming traffic flow.
Time efficiency, even thought is seldom covered in the literature, is one of the most important criteria for IDS: an IDS response time affects the entire organization, helping confining an attack in the best case, or making the IDS useless in the worst case. Usually there is a tradeoff between model complexity and performance, but using the already explained methods we were able to achieve the same predicting performances of computationally-heavy algorithms using lower computationally-costing algorithms, reducing the time needed for classification.
The report production phase is time consuming and it can reach almost 0.33% of the total classification time if half of the packets in the batch are attacks. Although this is a problem for time efficiency, we assumed that in a normal network, attacks are rarely received. Furthermore, switching to less consuming formats, such as .csv, could help mitigating this problem.

4. Project development

To develop this project we used python as main and only language. The selected IDE was Visual Studio Code and all the statistics are calculated on a Intel(R) Core(TM) Ultra 5 125U process.

4.1. UNSW-NB15 Dataset

The UNSW-NB15 dataset[10] is one of the few publicly available datasets for network-based anomalies and it immediately gained acknowledgments in literature. Among the available datasets, UNSW-NB15 is also one of the most recent one and most of the attacks it contains are still strongly used worldwide. The Cyber Range Lab of UNSW generated and captured using tcpdump 100GB of raw traffic. UNSW-NB15 contains a total of two million and 540,044 records. A partition of this databsed has been used as training and testing set. This partition is divided in 41 features and contains ten distinct categories:

1. Fuzzers;
2. Analysis;
3. Backdoors;
4. DoS;
5. Exploits;
6. Generic;
7. Recoinassance;
8. Shellcode;
9. Worms;
10. Normal.

Particularly Fuzzers are attacks born from bug hunting. Matter of fact, by feeding invalid data to a system, it can expose a bug and allow an attacker to exploit the vulnerability to gain unauthorized access, extract sensitive data, or disrupt services.
Analysis attacks exploits vulnerability found by analyzing a device's power consumption, network traffic, or database. Power consumption can be a crucial information to break cryptography algorithm and time used to elaborate a query in a database can expose hint on secret data.
Backdoors are piece of code, either maliciously or not, insert in a program to gain privileges access bypassing authentication and verifications steps. Backdoors are frequently used by systems administrator to accelerate their control routine on a system, but these shortcut are extremely dangerous and can allow attackers to gain full control of a whole system with a minim effort.
Denial of Service (DoS), are attacks with the aim of deny a particular service by inundates the network with a barrage of either valid or invalid messages. Usually to perform a denial of service, attackers use various devices to increase the rate of packet transmitted to the target server, in this case we talk about Distributed Denial of Service (DDoS).
With the term exploits we identify a method or piece of code that takes advantage of vulnerabilities

in software, applications, networks, operating systems, or hardware. Thanks to exploits, attackers can gain privileges access to a system or inject a malicious script in a legitimate software.
Generic attacks are a type of cyber attack that can be used to reconstruct information from a database, even if the attacker doesn't know the data or queries. Matter of fact, the majority of databases have different layer of secrecy and it's crucial to implement good security control to avoid attackers gat hint or even the whole secret data by submitting particular queries to the system.
As like in war, opponents information are crucial to perform a successful attacks, so attackers use Reconnaissance attack to gather as much information as possible on the target before performing a malicious attack.
Shellcode are small pieces of code, just like scripts, used as the payload in the exploitation of a software vulnerability. They usually, in turn, exploit vulnerability to allow another attack success.
Worms are one of the different type of malware. In particular, worms have the ability to auto-reproduce their code to infect different systems, all in autonomous. Attackers usually uses worms to infect different devices to perform a DDos attack.
With the category Normal we identify a normal action on the system, so it is not marked as an attack. The first nine categories contains attacks, while the last one contains the remaining traffic. This database was chosen from six different databases found in literature (KDD99-Cup, UNSW-NB15, CIC-IDS 2017, CSE-CIC-IDS 2018, BOT-IoT, CIDDS-001/CIDDS-002). The main reason why this database was selected are:

1. The database contains a collection of both manufactured and real-world traffic, making it realistic;
2. The database was collected using publicly available tools (tcpdump), which allows to recreate the collection process in a real network;
3. The Training and Testing sets were discovered to be sufficient to reach high precision even though the dimension of these datasets are contained. This leads to low training time, which could be crucial in case of a reset of the device following an attack to the IDS itself or a critical failure.

UNSW-NB15	
Attack Type	Percent
Analysis	0.82%
Backdoor	0.71%
Dos	4.97%
Exploits	13.52%
Fuzzers	7.36%
Generic	22.92%
Reconnaissance	4.25%
Shellcode	0.46%
Worms	0.05%
Normal	44.94%

Table 1: Composition of the DataFrame UNSW-NB15

4.2. Libraries

To develop this project we used the following Python libraries:

- Numpy
- Pandas
- Sklearn

In addition to these libraries, **Matplotlib** and **time** were used to produce the graphs and calculate the time-related statistics.

4.3. Employed metrics

To evaluate our model, we used the following metrics:
Accuracy measures how closely a model's predictions match the actual values.
The precision of a classifier measures the proportion of correctly predicted positive cases out of all cases predicted as positive.
The recall measures the proportion of correctly predicted positive cases out of all actual positive cases.
F_1 combines precision and recall into a single value, providing a balanced measure of how well the model performs.
FAR (False Alarm Rate) is a metric used to determine how many negatives were classified as positives. In our case, is the amount of attacks that were not discovered by the IDS.
Once the data have been classified, they can be divided into four groups:

- True positives (TP);
- True negatives (TN);
- False positives (FP);
- False negatives (FN).

The following is the calculation formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6}$$

$$FAR = \frac{FP}{FP + TN} \tag{7}$$

An additional metric was used: time. Classification time and detection time, which are the duration required to train the model and the duration required to categorize a sample using the trained model, are seldom found in literature. This metric is extremely relevant in our research and for IDSs. The main objective of an IDS is to achieve the highest precision with the lowest time consumption. Long classification and detection time indicates that the used model complexity is too high. This is a frequent problem for deep learning-based methods. We evaluated both classification and detection time to ensure that the proposed model is able to get as close as possible to real-time analysis and that a technical error, such as reset, would not drastically reduce the network security for a long time.

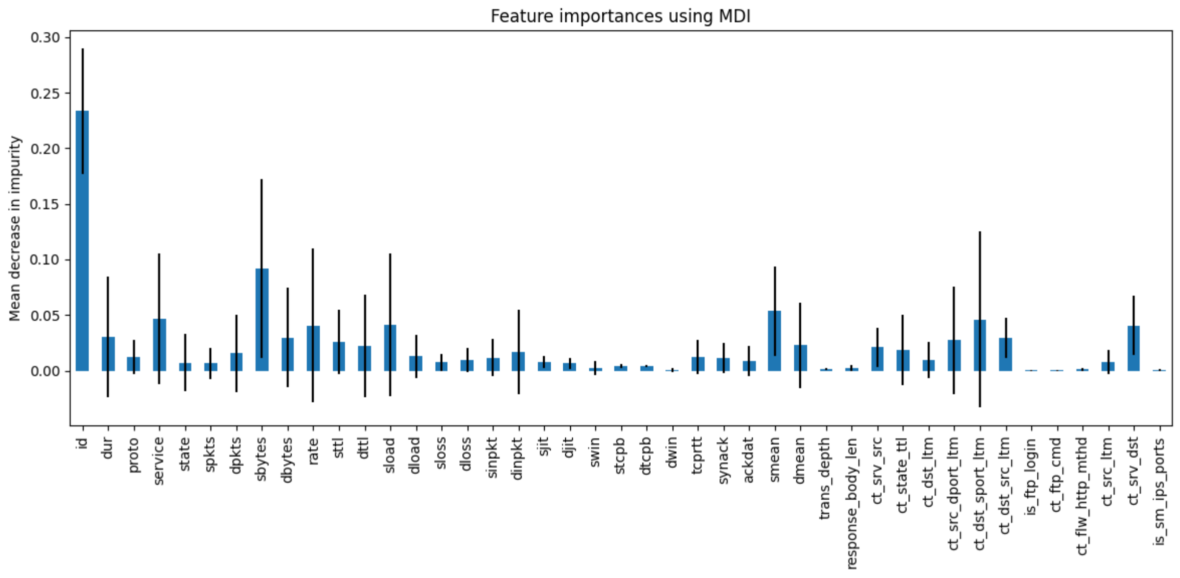
4.4. Data pre-processing functions and process

Decision Tree and Random Forest provide high accuracy, but can be prone to problems such as overfitting. To anticipate and solve these problems feature selection was used. The main advantages of using feature selection are:

- Avoiding the curse of dimensionality;
- Avoiding overfitting;
- Reducing the comprehensive time required for the model fitting and prediction;

- Facilitating the model interpretability.

We first calculated the feature importance using a property of the Random Forest classifier and produced the following graph: The lowest-valuing features were cut to reduce the model complexity and



improve the accuracy. After cutting the less meaningful features, text-including features were codified using a LabelEncoder offered by Sklearn. LabelEncoder codifies the values using a Numerical encoding, which does not require additional space nor increases the complexity of our model. This operation requires a different encoder for every textual feature. Another initial operation was the merge between the Testing and Training datasets of the UNSW-NB15 dataset. This was due to the unbalance between the two datasets (the Testing dataset contained more then double the instances of the Training dataset). The merged the two datasets, applied the pre-processing operations, and then divided them again, splitting them in a 70/30 ratio. To simulate a more realistic traffic flow, the Testing set was divided once again in several batches, each one of them containing approximately 2000 instances. All the provided time-regarding results are calculated using the mean of the times of the singular batches. Other encodings, such as One-hot, were considered. One-hot encoding specifically was considered be sure that the model did not value the labeled features as ordered sets. What the model showed was not only a slight decrease in the model predicting ability, but also a not indifferent increase in the classifying time. This increase was almost close to 100%. Taking in mind these considerations, One-hot encoding was immediately excluded.

4.5. Classification algorithms

Several classifiers were considered for the ensemble method. Decision Tree, Random Forest and MLP were chosen because they could achieve high accuracy in low time. KNN was later considered and the ensemble method that used the four of them actually surpassed the accuracy of the previous model with an advantage of 3%. The reason for its considered exclusion is due the time complexity needed for classification: the ensemble method that used the KNN was 50% slower. Because the wanted model had severe time constraints, the second model was put under severe scrutiny, but the accuracy advantage it provided was decisive for the final selection of this model.
If accuracy is desired, it is suggested to implement the ensemble method using four classifiers together. In a more complex system, the choice could be left to the operator or being based on the traffic rate analysis. Excluding the Deep Learning algorithm, the other classifiers were selected from literature. Time and accuracy were the main evaluators for the classifiers' selection. This is due the nature of the system. SVM was considered but the time complexity needed for the multi-feature classification far

exceeded the time constraints, so it was rejected. Linear Regression, specifically Binomial regression, was also considered. This classification algorithm could not achieve the same accuracy values of the chosen classifiers in the selected time slot. Decision Tree and Random Forest both achieved high accuracy in low time.

4.5.1 Decision Tree

A Decision Tree algorithm is a supervised learning method that uses a tree-like structure to classify data or make predictions. To compute decisions in the classifications, this algorithm consider the Shannon entropy of the possible classes before cluster the data considered. It also use another classification criteria called "Gini" that works like this: If a target is a classification outcome taking on values $0, 1, ..., K-1$, for node m, let

$$p_{mk} = \frac{1}{n_m} \sum I(y = k)$$

be the proportion of class k observations in node m. If m is a terminal node, the probability of the prediction for this region is set to p_{mk}. Said that Gini uses the following function:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

4.5.2 Random Forest

A Random Forest Classifier is a machine learning algorithm that uses the ensemble method to classify data with different Decision Trees.

4.5.3 MLP

A Multilayer Perceptron (MLP) is a forward neural network. MLP calculates the result based on a sequential convolution of the single layer results. In every layer there's at least one single Perceptron, which is the atomic unit of this neural network. A Perceptron[11] is a logic approximation of a human neural nerve. Perceptrons, as neurons do, take as input an array of binary inputs and produce a binary value. The time complexity[8] T of a MLP can be approximated following this formula:

$$T = ((n-1)/2)^2 \in O(n^2). \tag{8}$$

Given the fact, that the number of neurons n for a given problem can be regarded as a constant, the overall complexity is

$$O(n^2) = O(1). \tag{9}$$

where n would be the number of neurons that compose in the network.
To find the optimal MLP for our problem we evaluated the performance of several MLPs and chose the one that provided the best results on our dataset.

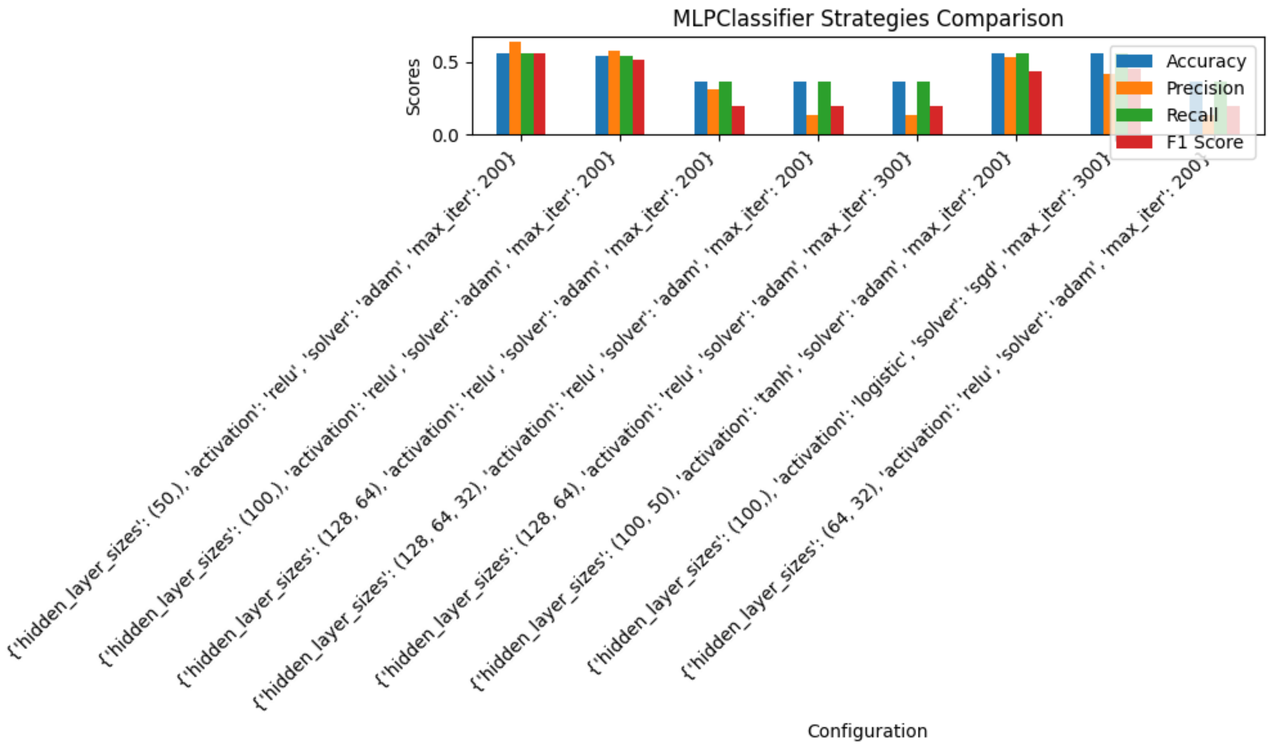


Figure 3: MLP comparison without StandardScaler

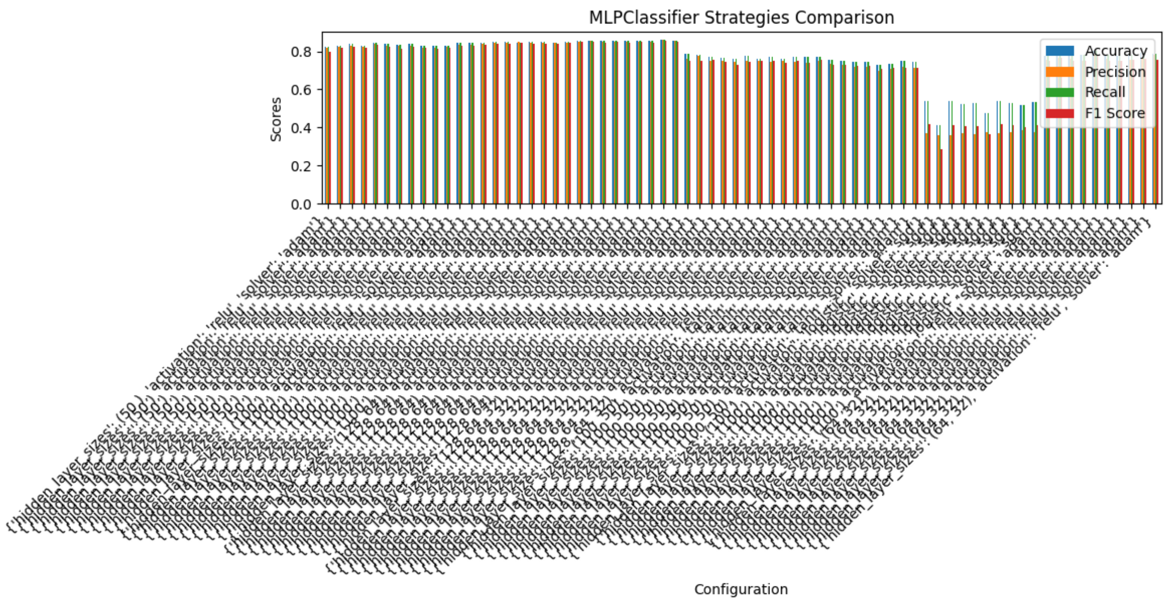


Figure 4: MLP comparison with StandardScaler

In the final application, before using the MLP, a StandardScaler was applied to reduce the model complexity and increase the precision of the MLP.
For the measuring, after applying the StandardScaler, the MLP were tested with these following strategies:

- First MLP
  1. hidden layer sizes: (50,);
  2. activation: 'relu';

    3. solver: 'adam';
- Second MLP
    1. hidden layer sizes: (100,);
    2. activation: 'relu';
    3. solver: 'adam';
- Third MLP
    1. hidden layer sizes: (128,64,);
    2. activation: 'relu';
    3. solver: 'adam';
- Forth MLP
    1. hidden layer sizes: (128,64,32,);
    2. activation: 'relu';
    3. solver: 'adam';
- Fifth MLP
    1. hidden layer sizes: (100,50,);
    2. activation: 'tanh';
    3. solver: 'adam';
- Sixth MLP
    1. hidden layer sizes: (100,);
    2. activation: 'logistic';
    3. solver: 'sgd';
- Seventh MLP
    1. hidden layer sizes: (64,32,);
    2. activation: 'relu';
    3. solver: 'adam';

For each of these strategies, a maximum number of iterations m_i was applied, with:

$$m_i \in [50, 100, 150, 200, 250, 300, 350, 400, 450, 500] \tag{10}$$

4.5.4 KNN

K-Nearest Neighbor (KNN) is a supervised learning classifier which uses proximity to make classifications or predictions about the grouping of an individual data point. KNN suffers greatly from the curse of dimensionality and it is highly sensible to feature scaling, so and Scaler was used to increase its classifying ability. KNN's performances were measured before the scaler application and after applying different scalers to find the optimal scaler and the optimal number of neighbors. In the end, MinMaxScaler was found to be the optimal scaler for our problem.

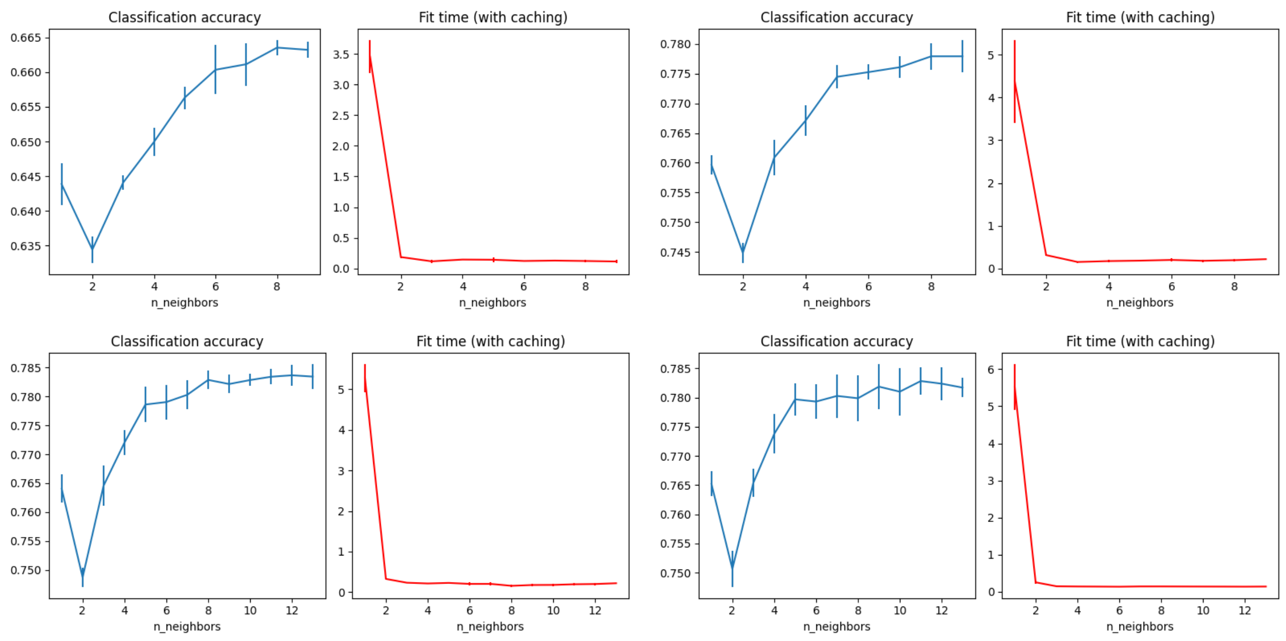


Figure 5: KNN without scaler, KNN with standard scaler, KNN with MinMaxScaler, and KNN with MaxAbsScaler

This approach also led to a reduction in the time used for fitting and classification.

4.6. Algorithm

Algorithm 1 IDS (Model)

```
 1: Import the UNSW-NB15 Dataset
 2: Testing set import
 3: Feature selection
 4: Label encoding
 5: Ensemble model fit
 6: for $Accuracy < Target$ do
 7:     Ensemble model testing
 8:     Accuracy Testing
 9: end for
10: Label decoding
11: Report production
12: Report saving
```

Algorithm 2 IDS (Possible System Proposal)

```
Import the UNSW-NB15 Dataset
Feature selection
Label encoding
Ensemble model fit
for $Accuracy < Target$ do
    Ensemble model predict
    Ensemble model testing
    Accuracy Testing
end for

while Incoming network traffic do
    while dim $<$ batchDimmension do
        batch $<=$ Traffic
    end while
    Label encoding of the batch
    Ensemble model predict
    Label decoding
    Report production
    Report saving
end while
```

5. Documents

In this section we will provide with the documents regarding the business model and the user experience. Other documents, such as case diagrams, state diagrams, or similar documents were not produced.

5.1. BPMN

In this section we will illustrate the BPMN model of the proposed IDS. This model concerns only the basic structure of the IDS, ignoring all the additional proposed features and the external processes. Specifically, the acquisition phase, which concerns another software, and the post-classification analysis, which requires additional care, are ignored. A more complex model will be provided in another section.

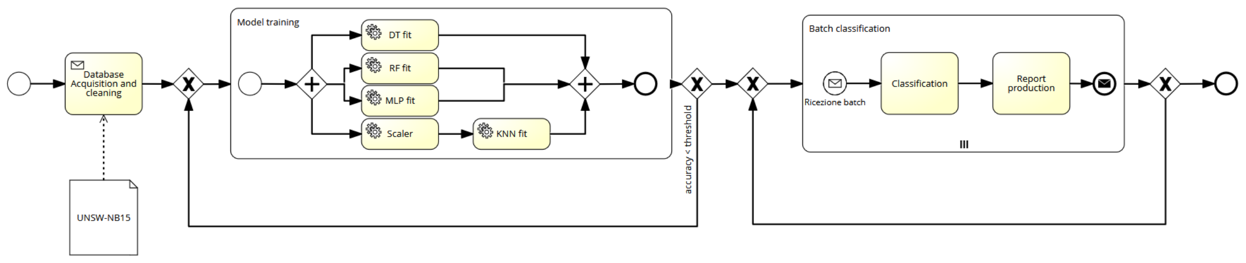


Figure 6: Basic IDS model

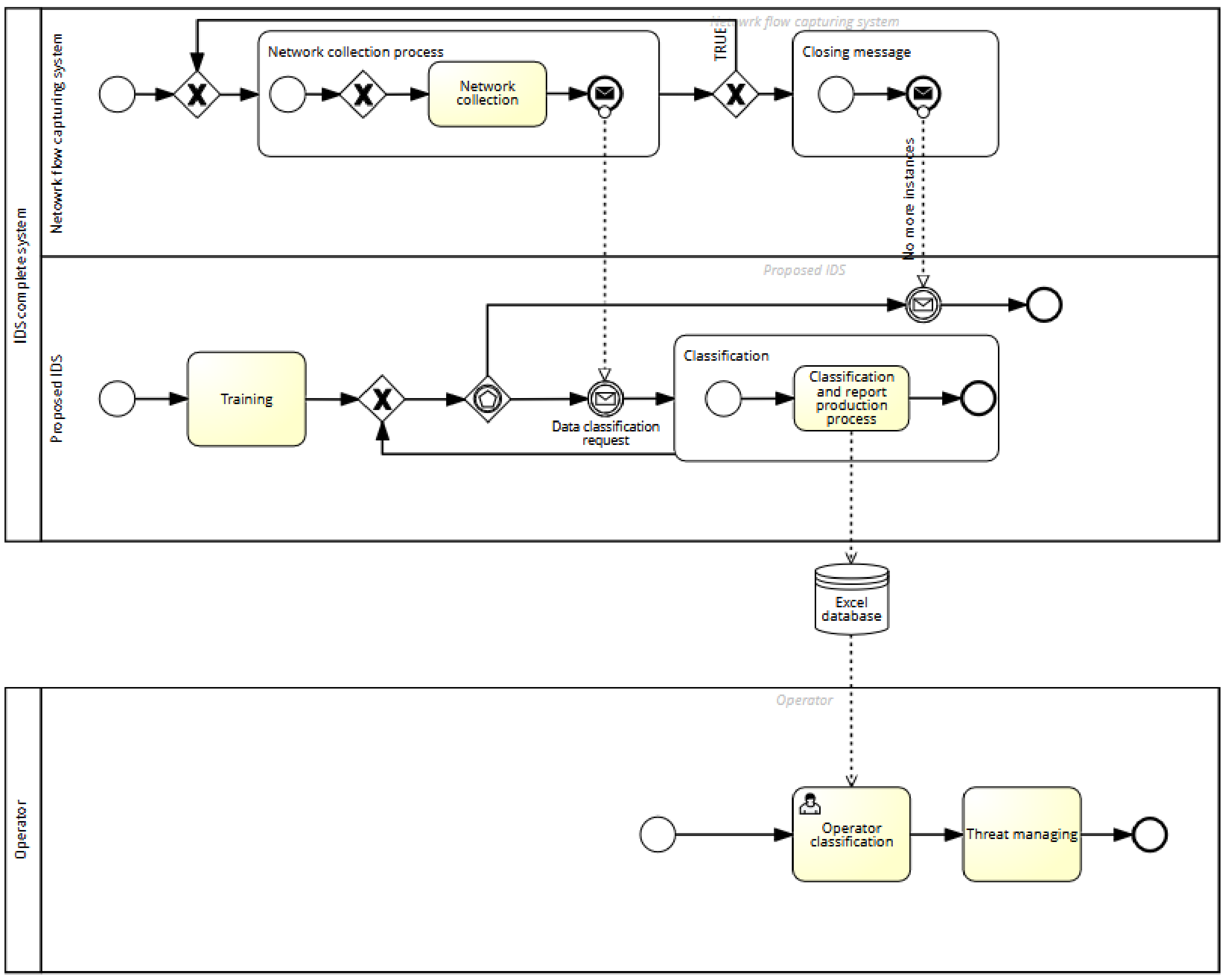
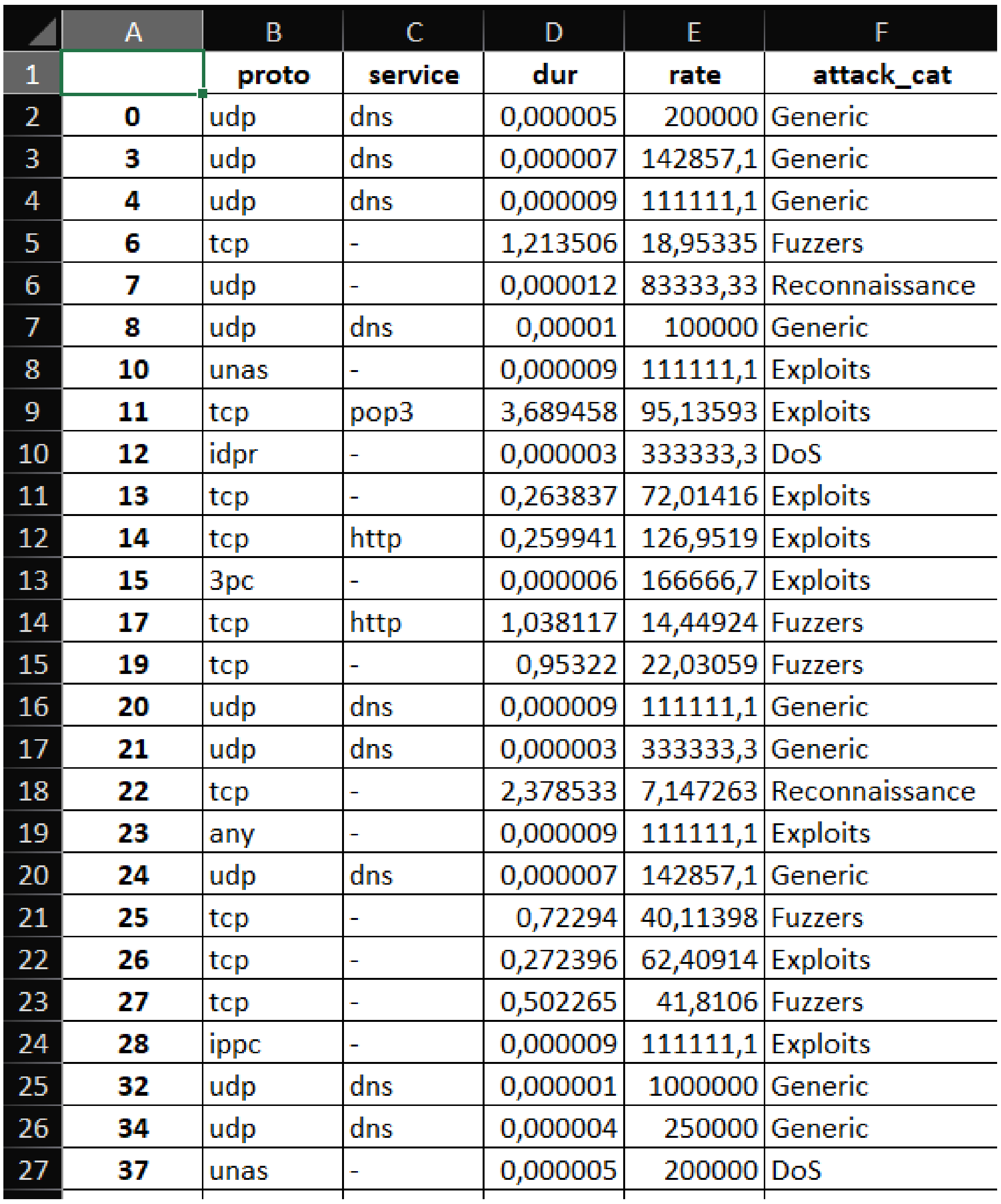


Figure 7: Contextualized IDS

5.2. UX

Regarding UX, the output of the proposed model can be found in a.xlsx. The following picture was taken from one of the results of a tested batch:



	A	B	C	D	E	F
1		proto	service	dur	rate	attack_cat
2	0	udp	dns	0,000005	200000	Generic
3	3	udp	dns	0,000007	142857,1	Generic
4	4	udp	dns	0,000009	111111,1	Generic
5	6	tcp	-	1,213506	18,95335	Fuzzers
6	7	udp	-	0,000012	83333,33	Reconnaissance
7	8	udp	dns	0,00001	100000	Generic
8	10	unas	-	0,000009	111111,1	Exploits
9	11	tcp	pop3	3,689458	95,13593	Exploits
10	12	idpr	-	0,000003	333333,3	DoS
11	13	tcp	-	0,263837	72,01416	Exploits
12	14	tcp	http	0,259941	126,9519	Exploits
13	15	3pc	-	0,000006	166666,7	Exploits
14	17	tcp	http	1,038117	14,44924	Fuzzers
15	19	tcp	-	0,95322	22,03059	Fuzzers
16	20	udp	dns	0,000009	111111,1	Generic
17	21	udp	dns	0,000003	333333,3	Generic
18	22	tcp	-	2,378533	7,147263	Reconnaissance
19	23	any	-	0,000009	111111,1	Exploits
20	24	udp	dns	0,000007	142857,1	Generic
21	25	tcp	-	0,72294	40,11398	Fuzzers
22	26	tcp	-	0,272396	62,40914	Exploits
23	27	tcp	-	0,502265	41,8106	Fuzzers
24	28	ippc	-	0,000009	111111,1	Exploits
25	32	udp	dns	0,000001	1000000	Generic
26	34	udp	dns	0,000004	250000	Generic
27	37	unas	-	0,000005	200000	DoS

Figure 8: Screenshot of a possible report from the IDS

In a more complex architecture, the sender's IP address and the receiver's IP address could be included in the report.

5.3. Archimate

We propose two distinct Archimate models. The first one is based on a NIDS that rests on a distinct architecture in the network, outside the client's physical devices. This first proposal is based on the idea that providing a physical architecture could mantain the same time-constraints that were considered, taking advantage of the use of GPUs or several cores. The second model is a software solution that is indipendent from the client's architecture. This solution, which is the implemented one, has no way to control the time efficiency for it depends on the number of cores in the physical machine.

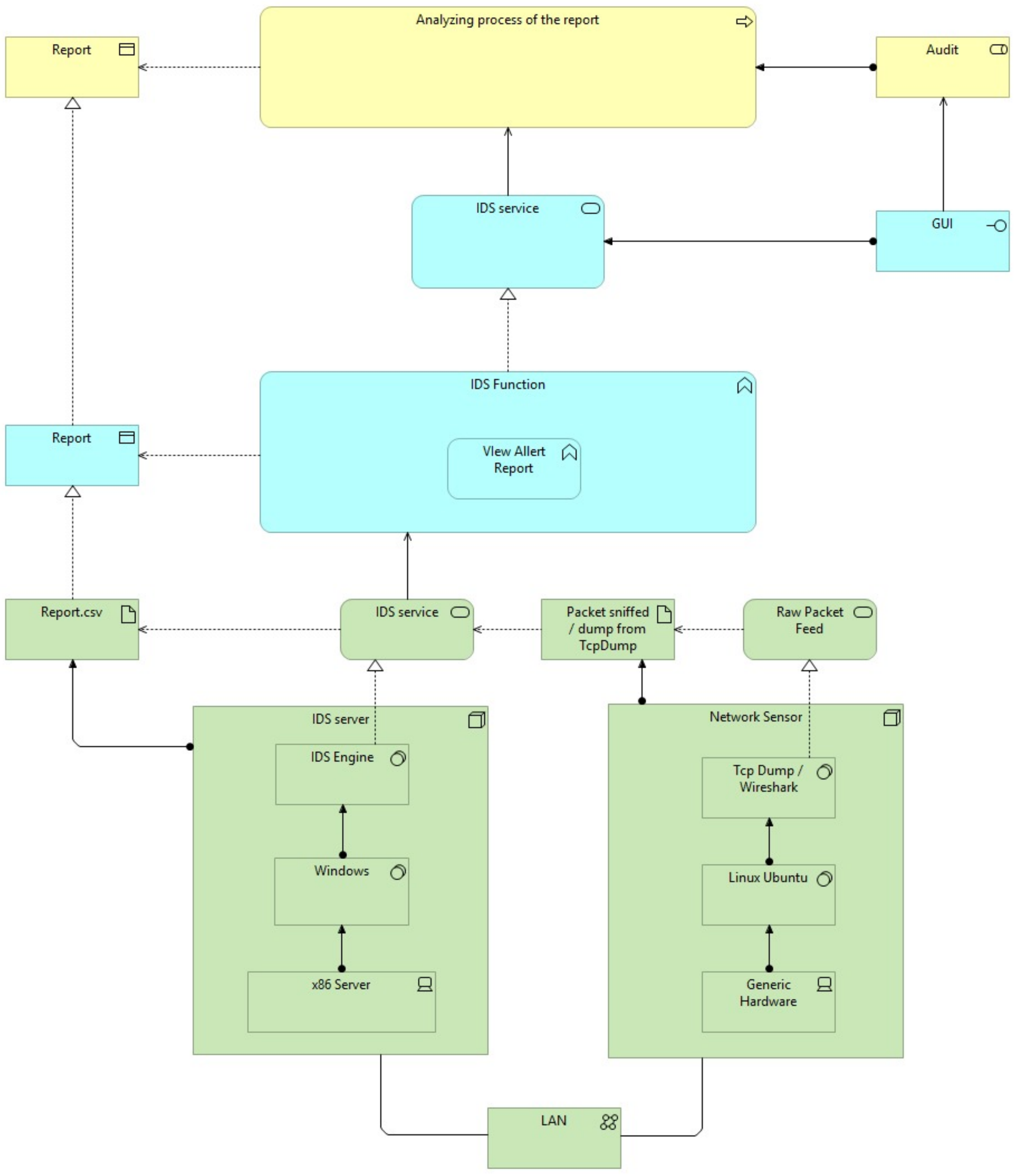


Figure 9: First model (hardware and software)

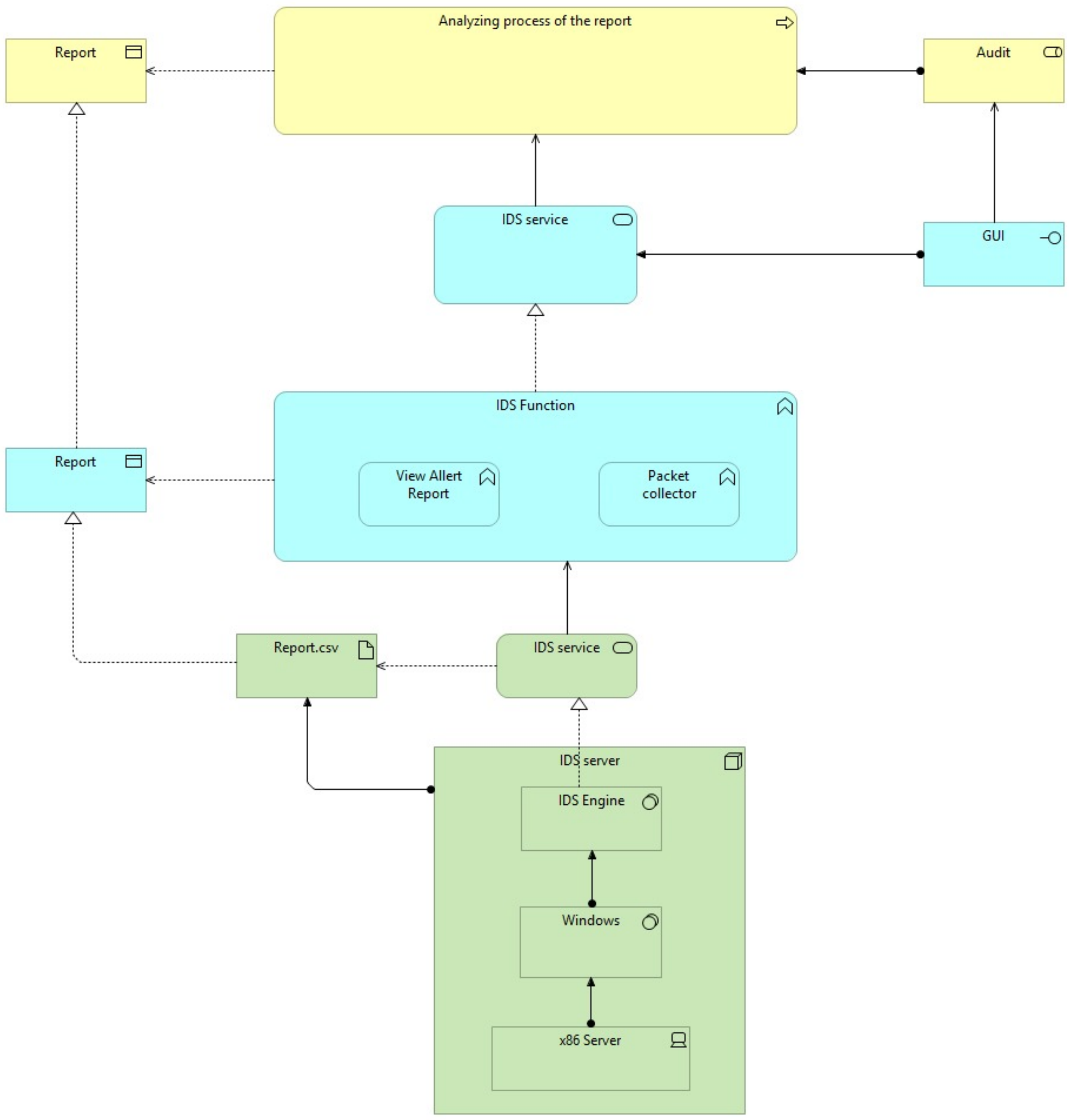


Figure 10: Second model (software only)

6. Results

Although some of the results cannot be shown with statistic, others can. In the first category we could find explainability, which was one of the cores of this research. On the latter, we have the already explained metrics. In this section are presented the results calculated using those metrics. The "Comparison of classifiers" graphs and the confusion matrices were obtaining using the entire testing dataset, while the first table was obtained testing both the optimal ensemble method (EN1) and the first ensemble method (EN2), on both binary (B) and multiclass (M) classification.

	DT	RF	MLP	KNN	EN1	EN2
Accuracy (B)	97.6440	96.5178	97.209	94.7844	97.7656	97.8834
Precision (B)	0.9766	0.9651	0.9720	0.9490	0.9776	0.9788
Recall (B)	0.97644	0.96517	0.97209	0.94784	0.97765	0.97883
F1-score (B)	0.97648	0.96518	0.97208	0.9481	0.9776	0.97884
FAR (B)	0.021	0.022	0.036	0.054	0.015	0.017
Fitting time(s) (B)	3.101364	55.27883	118.72924	0.19008	71.41491	111.06709
Classification Time(s) (B)	0.034102	0.92052	0.10433	19.21020	15.87057	17.43396
Accuracy (M)	84.4983	84.0634	84.5423	79.3540	84.9501	84.9760
Precision (M)	0.8410	0.8447	0.8472	0.7806	0.8441	0.8462
Recall (M)	0.8449	0.8406	0.8454	0.7935	0.8495	0.8497
F1-score (M)	0.8305	0.8149	0.8414	0.7835	0.8309	0.8339
FAR (M)	0.027	0.044	0.040	0.058	0.029	0.026
Fitting time(s) (M)	6.5613	81.8248	179.2561	0.2057	72.7085	171.8438
Classification Time(s) (M)	0.0523	1.4056	0.1741	23.0310	20.5895	21.2562
Reporting Time(s)	0.25	0.25	0.25	0.25	0.25	0.25

Table 2: Performance statistics

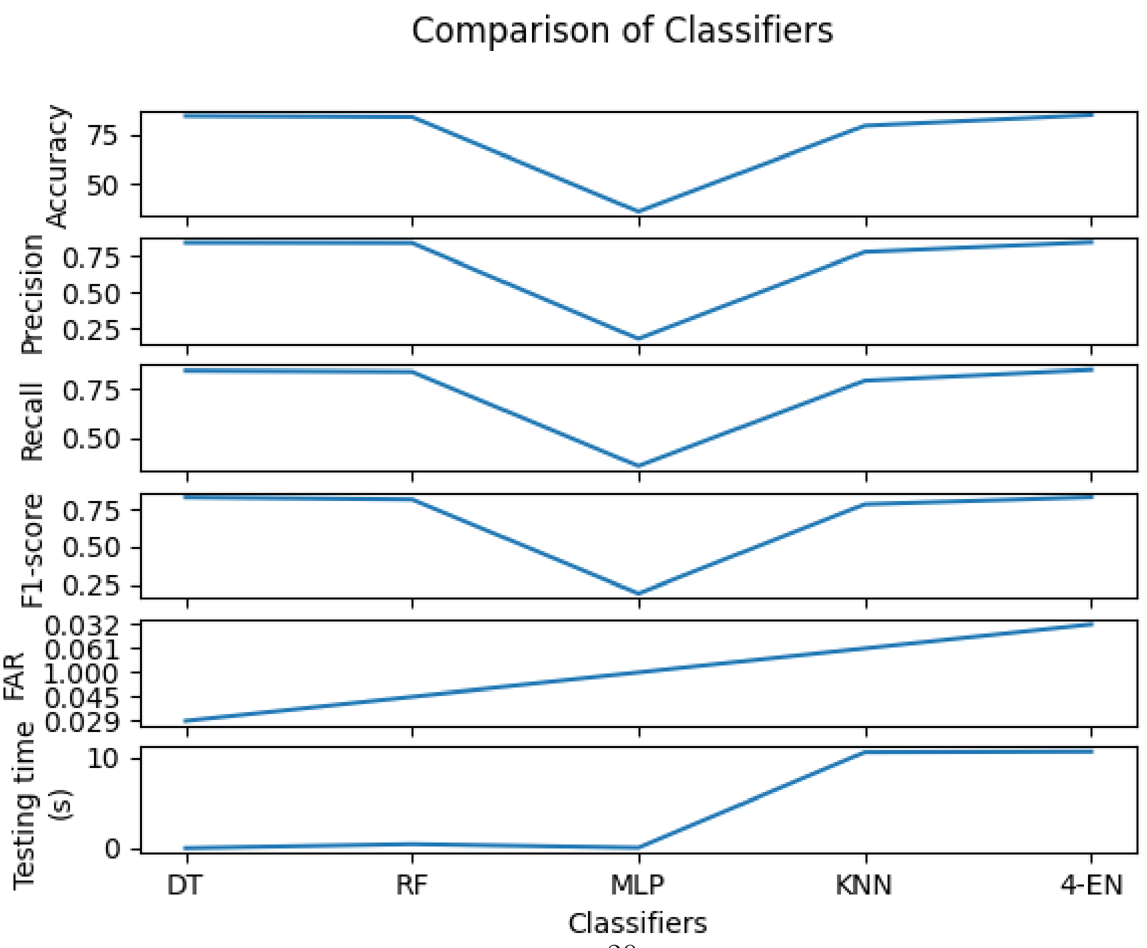


Figure 11: Accuracy, Precision, Recall, F1-score, False positive rate, and classification time of the classifiers

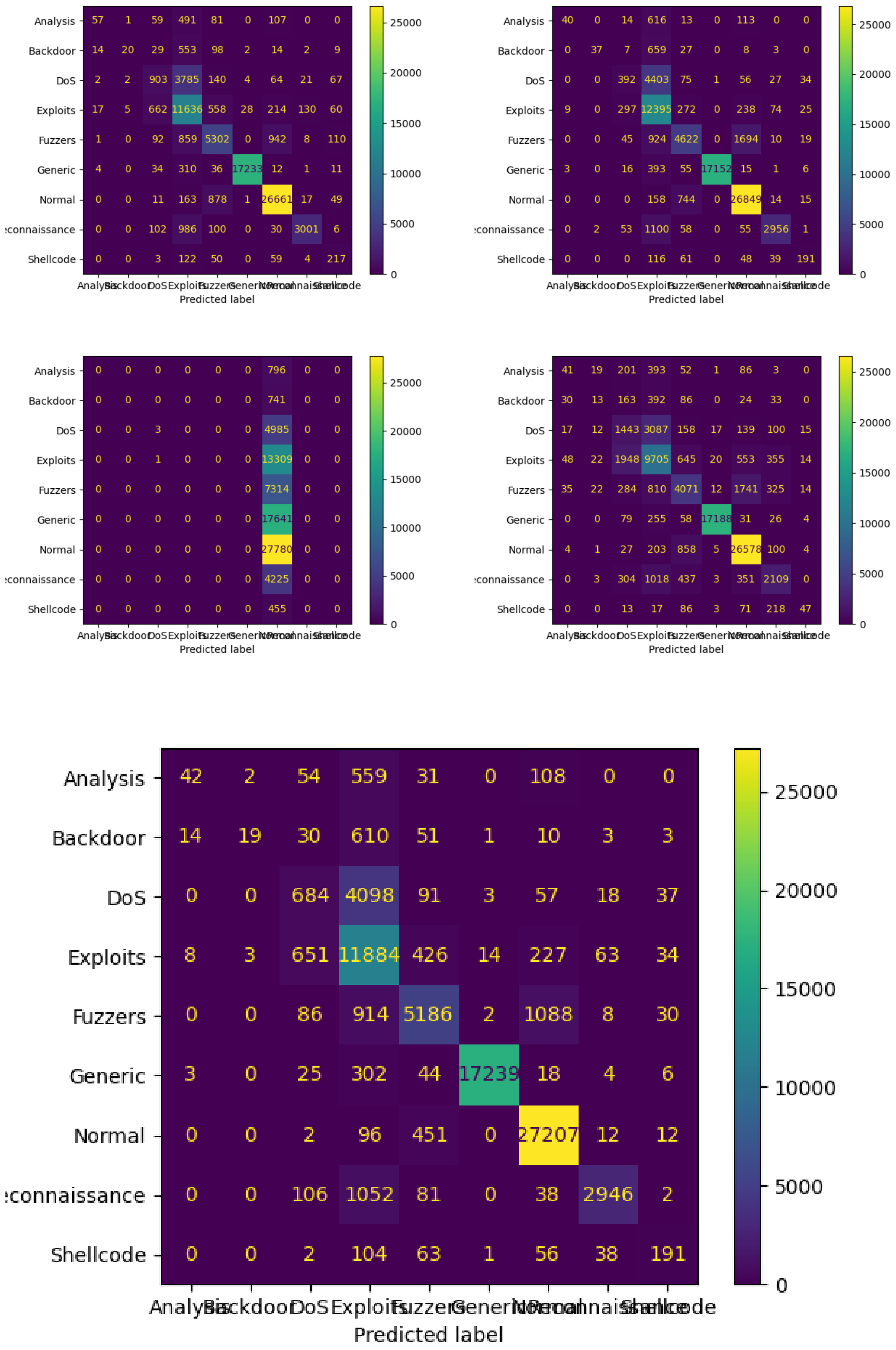


Figure 12: Confusion matrices of: DT(Upper left), RF(Upper right), MLP(Center left), KNN (Center right), The ensemble model(Bottom) 21

It is clear that the MLP strongly under performs the other classifiers. We also found a semi-optimal hyperparameter configuration to increase the performances of the MLP while maintaining the the same temporal cost. The best solution we found was to use a standard scaler followed by a MLP with just a hidden layer containing 100 Perceptrons. In addition, the newly implemented MLP has a maximum number of possible iterations set to 100. With this change it is possible to achieve the following statistics:

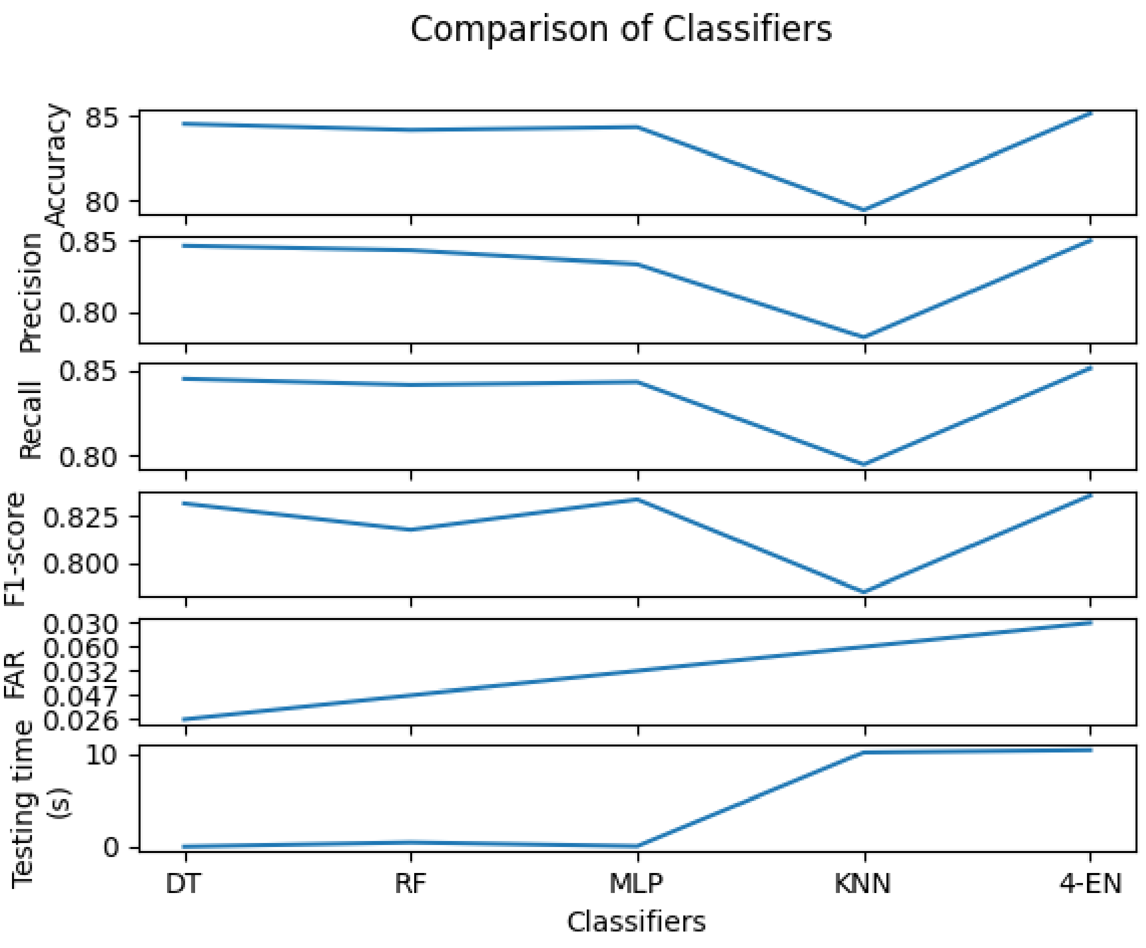
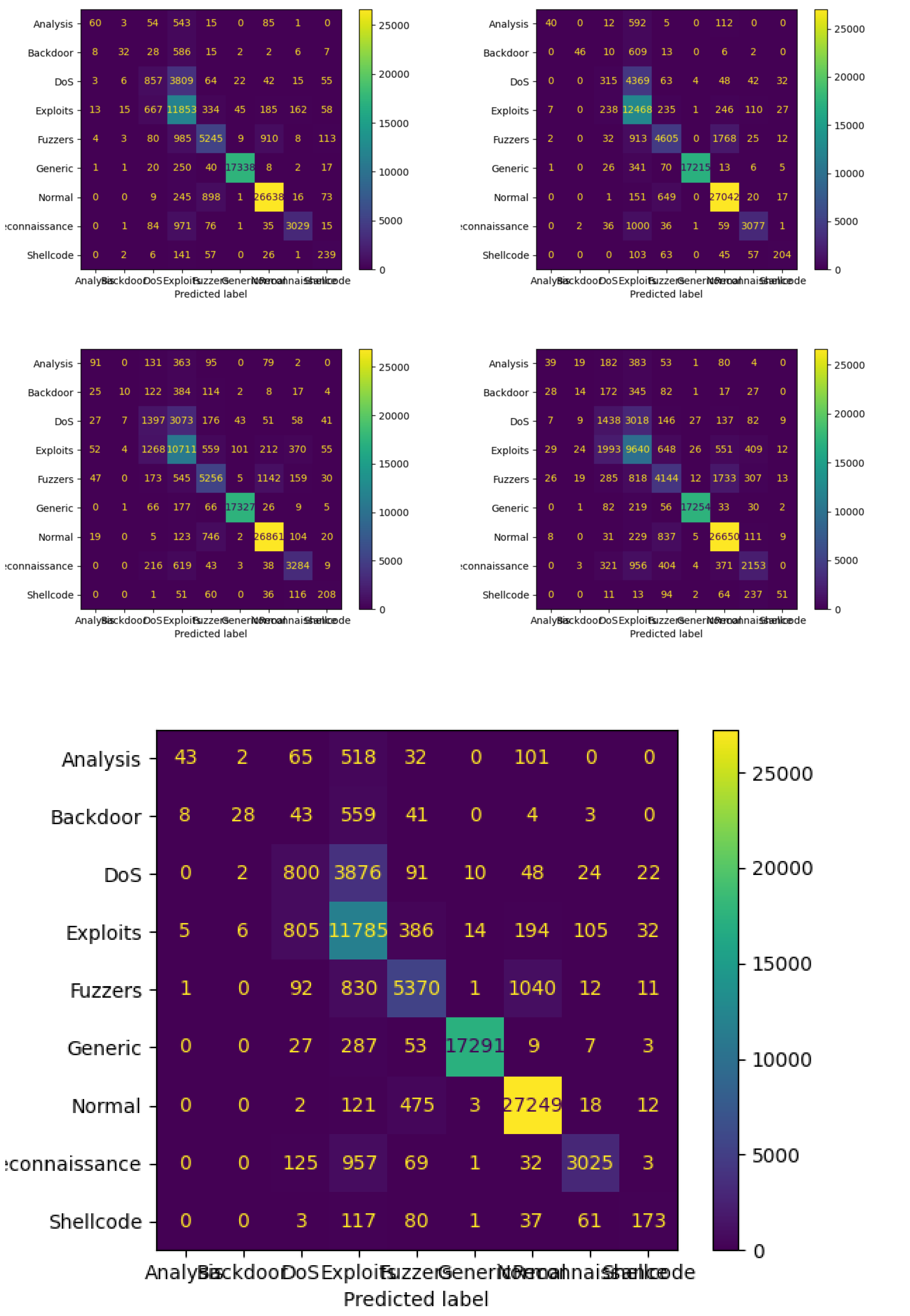


Figure 13: Accuracy, Precision, Recall, F1-score, False positive rate, and classification time of the classifiers



	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode
Analysis	60	3	54	543	15	0	85	1	0
Backdoor	8	32	28	586	15	2	2	6	7
DoS	3	6	857	3809	64	22	42	15	55
Exploits	13	15	667	11853	334	45	185	162	58
Fuzzers	4	3	80	985	5245	9	910	8	113
Generic	1	1	20	250	40	17338	8	2	17
Normal	0	0	9	245	898	1	26638	16	73
econnaissance	0	1	84	971	76	1	35	3029	15
Shellcode	0	2	6	141	57	0	26	1	239

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode
Analysis	40	0	12	592	5	0	112	0	0
Backdoor	0	46	10	609	13	0	6	2	0
DoS	0	0	315	4369	63	4	48	42	32
Exploits	7	0	238	12468	235	1	246	110	27
Fuzzers	2	0	32	913	4605	0	1768	25	12
Generic	1	0	26	341	70	17215	13	6	5
Normal	0	0	1	151	649	0	27042	20	17
econnaissance	0	2	36	1000	36	1	59	3077	1
Shellcode	0	0	0	103	63	0	45	57	204

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode
Analysis	91	0	131	363	95	0	79	2	0
Backdoor	25	10	122	384	114	2	8	17	4
DoS	27	7	1397	3073	176	43	51	58	41
Exploits	52	4	1268	10711	559	101	212	370	55
Fuzzers	47	0	173	545	5256	5	1142	159	30
Generic	0	1	66	177	66	17327	26	9	5
Normal	19	0	5	123	746	2	26861	104	20
econnaissance	0	0	216	619	43	3	38	3284	9
Shellcode	0	0	1	51	60	0	36	116	208

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode
Analysis	39	19	182	383	53	1	80	4	0
Backdoor	28	14	172	345	82	1	17	27	0
DoS	7	9	1438	3018	146	27	137	82	9
Exploits	29	24	1993	9640	648	26	551	409	12
Fuzzers	26	19	285	818	4144	12	1733	307	13
Generic	0	1	82	219	56	17254	33	30	2
Normal	8	0	31	229	837	5	26650	111	9
econnaissance	0	3	321	956	404	4	371	2153	0
Shellcode	0	0	11	13	94	2	64	237	51

	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode
Analysis	43	2	65	518	32	0	101	0	0
Backdoor	8	28	43	559	41	0	4	3	0
DoS	0	2	800	3876	91	10	48	24	22
Exploits	5	6	805	11785	386	14	194	105	32
Fuzzers	1	0	92	830	5370	1	1040	12	11
Generic	0	0	27	287	53	17291	9	7	3
Normal	0	0	2	121	475	3	27249	18	12
econnaissance	0	0	125	957	69	1	32	3025	3
Shellcode	0	0	3	117	80	1	37	61	173

Figure 14: Confusion matrices of: DT(Upper left), RF(Upper right), MLP(Center left), KNN (Center right), The ensemble model(Bottom)

The downside to the implementation of the second MLP can be found in the training time. The training cost increases greatly (at least 50%).
We also explored binary classification in more detail. As the next picture will show, the ensemble method outperforms the selected classifiers:

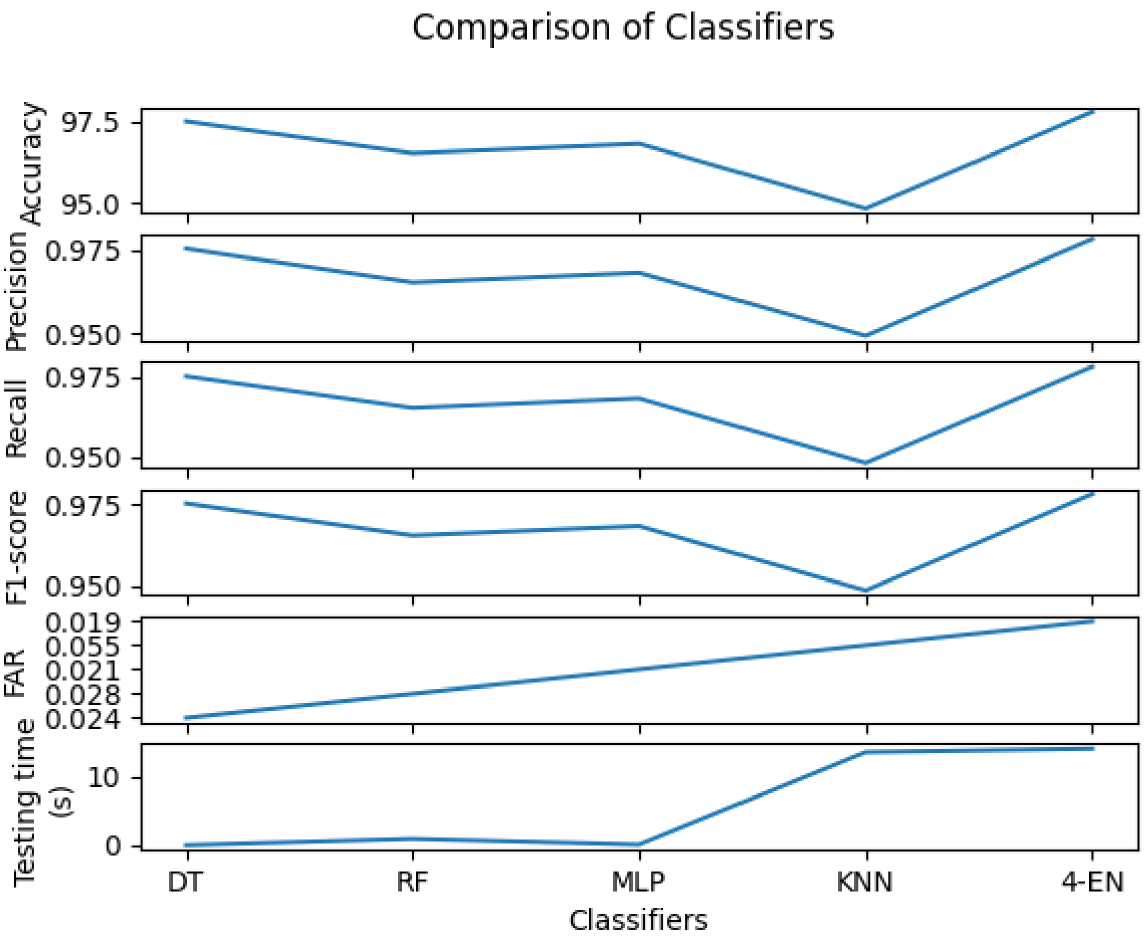


Figure 15: Binary classification comparison

In conclusion, the used Ensemble method performs better than the individual classifiers, outperforms each one of them in the False Alarm Rate (FAR), and is as time-costly as the worst of them.
We then present the ROC curve of the classifiers:

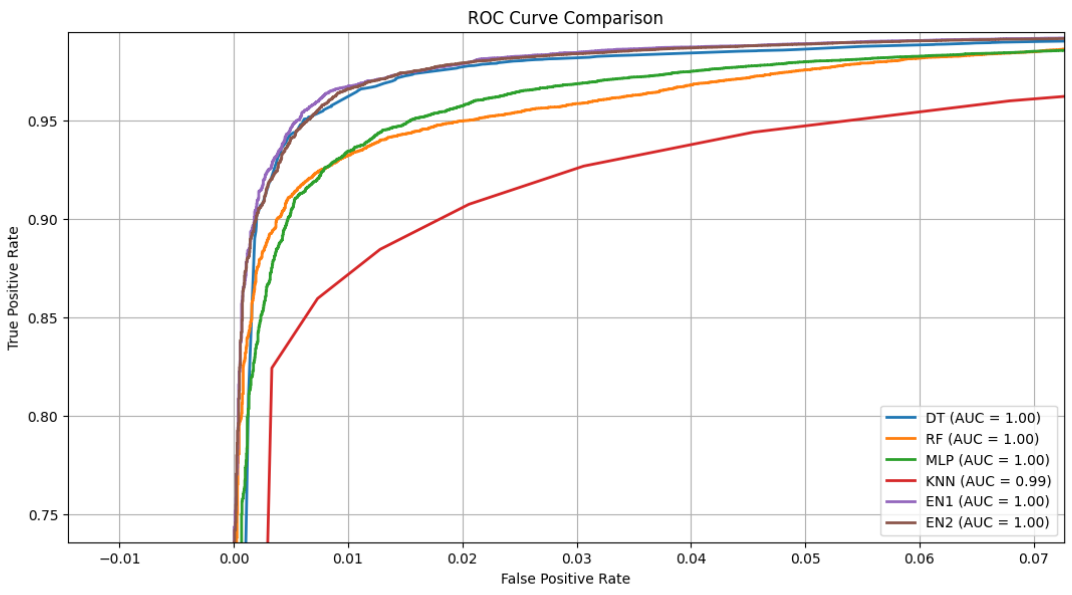


Figure 16: Binary ROC curve

Regarding Multi-class classification, we now present the ROC curve obtained using the OvR (One-vs-Rest) strategy:

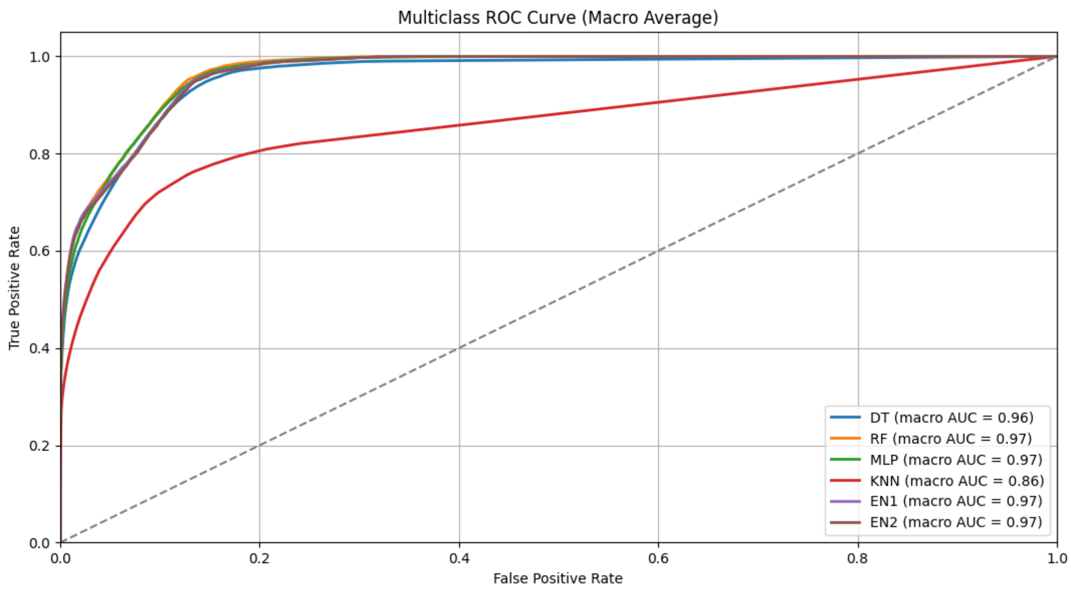


Figure 17: OvR ROC curve

7. Conclusions

Intrusion Detection Systems are vital for enterprises' security. While literature has been focused on signature-based systems for many years, anomaly-based IDSs are starting to appear in literature. These systems usually achieves lower accuracies and higher false alarm rates, but are more agile and can discover unknown attacks. Another downside of these systems is the computational complexity, which is usually very costly, but finding the optimal model can help reducing the cost.

A state-of-the-art model was derived from the literature and implemented, achieving high precision in low time by combining several classifiers to achieve high precision in binary classification of malicious traffic and high precision, even though lower than the one obtained for binary classification, in multi-class classification for attack-type classification. Accuracy, FAR, and time were the most valuable metrics in this research and were both deeply explored and achieved.
Explainability was explored in the final output of the model, which consists in a .xlsx format file containing five features (Protocol, Service, Duration, Rate, Attack category). This format can be easily changed both in the file format and in the features. Regarding the format, a .csv was also explored to reduce the report production time, but we chose to use a .xlsx to improve readability. Regarding the reported features, the IP addresses of the sender and the receiver could be added easily, for they are collected by tcpdump.
Future researches should try to improve the Deep Learning model and the balancing of the hyperparameters. Future researches should also reduce the comprehensive time by implementing more time-efficient classifiers (XGBoost Trees,...) or using CUDA-based solution. The deep-learning algorithm, which is a bottleneck regarding classification time, should also be researched with more attention. We researched the optimal MLP for this problem, but other deep-learning algorithms could possibly maintain the same performances reducing the time complexity. This is an open debate in literature and several propositions were made, still none of them explored time complexity and we believe it should be an essential metric. The proposed model could be implemented with low effort as a server solution or a on-device program.
There's something that we'd like to point out: this model, which is the classificatory motor of a IDS, is only a part of a larger system. To implement this solution in reality, there would be a need of a traffic collector and of a UI system that allows the auditor to visualize the excel files that the model produces. This point was marked several times in the paper and it is crucial to the understanding of the model as a whole.
This report presents a network-based, hybrid-based, passive IDS. We reviewed current works and, in accordance with literature, we proposed a system that uses both machine learning and deep learning to classify real-world traffic in an efficient, time-effective, and precise manner. Furthermore, we discussed current challenges and future research directions.

References

[1] V. Piuri A. Genovese. Anomaly-based intrusion detection system for ddos attack with deep learning techniques. *IRIS, ARI UniMI*, 2023.

[2] Pierpaolo Dini, Abdussalam Elhanashi, Andrea Begni, Sergio Saponara, Qinghe Zheng, and Kaouther Gasmi. Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity. *Applied Sciences*, 13(13), 2023.

[3] Vanlalruata Hnamte and Jamal Hussain. Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. *Telematics and Informatics Reports*, 10:100053, 2023.

[4] $(CS)^2$ AI-KPMG Control System Cybersecurity Annual Report.

[5] Namo Jain, Mukund Singh, Kuldeep Chaurasia, and Gobinath Ravindran. Intrusion detection system using ensemble techinque. pages 1–5, 09 2023.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[8] Björn-Elmar Macek. What is the time complexity of multilayer perceptron (mlp) and other neural networks?, 02 2014.

[9] Marija Milošević, Vladimir Ciric, and Ivan Milentijevic. Network intrusion detection using weighted voting ensemble deep learning model. 06 2024.

[10] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015.

[11] Frank Rosenblatt. The perceptron: A perceiving and recognizing automaton. Report, Project PARA, Cornell Aeronautical Laboratory, January 1957.

[12] Yoheswari S. *Optimized Intrusion Detection Model for identifying known and innovative cyber attacks using Support Vector Machine (SVM) algotihms.* Journal of Science Technology and Research (JSTAR), 2024.

[13] Tommaso Sacchetti. A study on controller area network deep-learning intrusion detection systems. *POLITesi*, 2023.

[14] S. Sridevi, R Prabha, K. Narsimha Reddy, K. M. Monica, G. A. Senthil, and M. Razmah. Network intrusion detection system using supervised learning based voting classifier. *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pages 01–06, 2022.

[15] Cisco Systems. 2024 global threat analysis report. *POLITesi*, pages 4–6, 2024.