LUISS

Department of Business and Management

Master of Science in Data Science and Management

A Cybersecurity Application of Data Science: Intrusion Detection Systems and Data Poisoning

Supervisor:

Prof. Alessio Martino

Co Supervisor:

Prof. Paolo Spagnoletti

Davide Rosatelli

Academic year 2022/2023

Acknowledgements

I would like to express my sincere gratitude to all those who have supported me through my university journey and in the writing of this thesis. First and foremost, I am thankful to my Supervisor, Prof. Alessio Martino, and my Co-Supervisor, Prof. Paolo Spagnoletti, for their guidance, expertise, and invaluable feedback, which have been of paramount importance in the completion of this work, without which it would not have had the form and completeness it has reached.

I would also like to extend my gratitude to all the professors who have guided me throughout my university journey, particularly Prof. Giuseppe Italiano, for his dedication and passion in the faculties of Management and Computer Science, as well as Data Science and Management. Furthermore, I would like to thank the entire university community, my fellow colleagues who actively participated in all the projects undertaken, from whom I have often learned a lot, and to everyone who works every day to contribute to making Luiss the stimulating environment that it is.

I am incredibly grateful to my entire family for the love, support and trust that I have always received. Whichever goal I may achieve in the future, it will always be largely thanks to the education provided to me and the sacrifices made by my parents, for this reason, I will never cease to express my gratitude. I would also like to say thanks and do my best wishes to Caterina, hoping that she will always has courage, desire to achieve, and luck on her side.

I would like to express my gratitude to all my friends, especially the YP, with whom I have shared so much over the years, and whose presence has been essential in both the sad and happy moments. Lastly, I would like to thank Alice, whose presence has made my life more beautiful, making me feel loved every day.

Table of content:

1. Introduction

In recent years the world is going through the so-called Fourth Industrial Revolution, commonly known also as Industry 4.0, which sees the advances of digitalization as its driving force. Information technology has become a prerogative in every field, and it is enabling year after year to make all kinds of work easier and more efficient. Artificial Intelligence (AI) has become a widely discussed topic, leaving even insiders wondering how far its applications can go and how much it will change the labour world known today. These prominent AI systems often rely on the field of data science, which through complex statistical algorithms manage to learn from these data and work out answers to the tasks they are asked. It is not surprising that data science has become one of the most popular and in-demand fields, offering innovative solutions to an innumerable amount of problems.

As new technologies advances and digitalization pervades every aspect of our lives, as it is obvious, the potential risks to face in the digital world have also increased, and this is why one of the other fields that is having great interest is cybersecurity. There are many aspects to keep under control, from the correct functioning of technological infrastructures to the protection of personal data, and in some of these tasks, data-driven systems such as machine learning algorithms can play a crucial role.

Organizations are just starting to see the potential advantages of these technologies. Consequently, machine learning and artificial intelligence are receiving increasing attention in the field of cybersecurity. Data-driven algorithms can be used to detect, identify, and react to threats, enabling a considerably faster response time, even real-time, and reducing the possibility of significant harm. AI-enabled systems can spot trends in data and predict potential outcomes, enabling proactive threat detection and the development of effective countermeasures. For instance, AI-based systems may search for system and network vulnerabilities as well as identify user behaviour anomalies that may be signs of criminal activity.

In addition to detection and identification, machine learning and AI can be used to automate several of the time-consuming and laborious cybersecurity processes, such as security incident response. Processes can be automated to increase efficiency, cut costs, and free up resources for more demanding tasks. AI-powered systems can be used to examine historical events in order to find new attack vectors and trends, serving as the basis to new strategies and countermeasures. Data science can also help an organization's security posture overall by giving it more precise information and making it simpler to administer. Moreover, AI-based systems can generate reports, and warn businesses about potential hazards or harmful conduct.

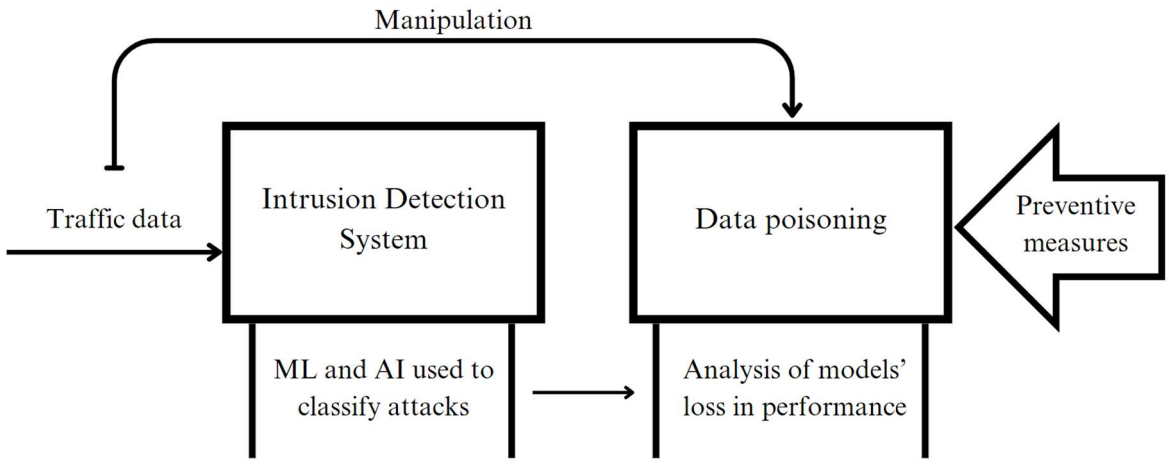
This work examines a possible scenario where data science and cybersecurity are interconnected. A dataset of network traffic will be worked on using machine learning algorithms with the goal of developing an Intrusion Detection System (IDS). An IDS is a hardware or software technology designed to alert network administrators when potential threats or attacks are detected, assisting them in responding more quickly to prevent or minimize damages. After the system for detecting suspicious traffic has been developed, it will be tested performing a simulation of data poisoning, a cyberattack involving deliberate feeding of machine learning algorithms with malicious data, in an attempt to invalidate the models.

This scenario is particularly relevant to study, due to its high likelihood of occurrence and its applicability to the Internet of Things (IoT) area: IoT defines a vast network of physical devices, including computers, vehicles, appliances, and other items that communicate one another and with other systems via the internet, though physical sensors and data exchange. These devices might be anything, from wearables and smart homes to smart cities and industrial equipment. As the prevalence of IoT devices increases, so does the need for robust cybersecurity measures to defend against cyberattacks. As these devices communicate among them through the exchange of huge quantity of data in a network, one of the most popular ways though which IoT systems defend themselves are intrusion detection systems, and since many of these systems are built through machine learning and artificial intelligence, even though their purpose is precisely to identify anomalies, they are susceptible to data poisoning.

This study aims to assess the effectiveness of machine learning in detecting cyber-attacks and its resilience to another type of cyber vulnerability. By providing some examples in specific environments, such as the IoT descripted above, it will be then feasible to draw the required conclusions to identify practical solutions to implement, given the importance of the current network of these types of devices and the approaching threat. A general scheme of the different points included in this work are represented in *Figure 1*.

In order to provide a basic framework for the creation of an IDS by machine learning and test its resistance to malicious data infiltration, this paper will focus on investigating the aforementioned scenario. It will be structured in several stages, carrying out the practical work on Python and analysing the outcomes with the appropriate measures; in addition, in the final sections of the work different methods of prevention against malicious data modification will be studied, to analyse their advantages and limitations. With the expectation of positive findings, this work will be relevant to a wide range of increasingly typical circumstances in contemporary cybersecurity.

Figure 1: Conceptual framework



1.1. Summary

In a world currently undergoing pervasive digital transformation across various industries, Artificial Intelligence (AI), and the broader domain of data science, have emerged as highly requested disciplines, as they offer innovative solutions to a multitude of challenges. Concurrently, with the rapid technological advancements and widespread digitalization, the associated risks in the digital world have expanded significantly, and to effectively manage the critical aspects of these novel systems, the field of cybersecurity has also gained substantial importance. Data-driven systems, such as machine learning algorithms, can assume a pivotal role in addressing various cybersecurity tasks, such as attacks identification, proactive threat response, and process automation.

This study delves into a plausible scenario in which data science and cybersecurity intersect, leveraging a dataset of network traffic to construct an Intrusion Detection System (IDS) designed to identify potential cyber threats, achievable through classifications. The system's resilience is then tested against diverse variations of the dataset, simulating data poisoning, an attack involving the deliberate injection of malevolent data to undermine machine learning models. This scenario holds relevance within the context of the Internet of Things (IoT), a network of interconnected devices that frequently employs IDS technology for self-defence, while simultaneously facing vulnerability to data poisoning attacks. Subsequently, after conducting practical simulations in Python, this study explores potential preventive measures against data poisoning.

The second chapter of this work conducts a comprehensive review of technical literature concerning IDSs and data poisoning, aiming to provide a comprehensive understanding of the current state-of-

the-art of these systems and attacks. The analysis of pertinent research papers, including reviews and surveys, reveals that contemporary IDSs exhibit high efficiency in monitoring network traffic and detecting malicious activities, frequently employing complex algorithms, such as custom ensemble models and deep learning techniques. One review focuses on scenarios involving the deployment of IDSs within IoT environments, emphasizing their appropriateness due to the distributed nature of these infrastructures, as also showed in two articles exploring the branches of mobile devices and critical infrastructures, both closely tied to IoT, and susceptible to elevated threat exposure. A common challenge highlighted in the development of machine learning based IDSs is the imperative need for quality data, and for this reason the UNSW-NB15 dataset, a repository of network traffic encompassing both normal and malicious activities, developed for research purposes, has gained substantial popularity. The literature review not only addresses the dataset's features but also explores some of its noteworthy applications.

The subsequent section of the literature review examines data poisoning attacks. Here, the analysed articles encompass surveys and studies explaining various methods through which these attacks can be executed, highlighting their danger as they can be applied to any algorithm, causing them to make harmful decisions. These works also explores several preventive measures against data poisoning, which are further explored in the subsequent chapter titled "Preventive Measures". Furthermore, some of these studies present again scenarios involving IoT, elucidating how these networks are particularly susceptible to adversarial attacks due to the presence of multiple vulnerable data access points.

Having framed the literature pertaining to these two facets of cybersecurity and identified some primary areas for further exploration, the third chapter of this work defines its principal objectives, encapsulated in the following research questions:

How good are the performances of machine learning and deep learning models in an IDS classification task? How much does their performance change between just detecting suspicious activity or recognising the type of attack in progress?

What is the impact of a simple data poisoning simulation on these models? How much do their performances change according to the different methods and the several intensity levels of poisoning?

What preventive measures against data poisoning can be implemented in the construction of this type of IDS? What are their advantages and limitations?

The “Methods” chapter reviews in detail the code essential for conducting this study, categorizing it into various sections that elucidate the different procedural steps. Initially, an explanation of the dataset is provided, which includes a simulation of realistic network traffic, encompassing numerous

simulated attacks, from which are derived numerous features, covering fundamental traffic information, content-related characteristics, network flow attributes, and two label columns utilized for binary and multiclass classification tasks. The original dataset is notably large, and for the purposes of this study, the train and test splits provided by the dataset's authors were employed, following some basic preprocessing procedures.

The preprocessing steps in the Python notebook include the transformation of all instance values into finite values and the conversion of categorical variables into numerical equivalents through ordinal encoding. Subsequently, a correlation matrix in the form of a heatmap is generated to identify factors most correlated with the target variables. The two originally provided files are merged to create new train and test splits, with labels stratified to mitigate imbalances among observations across different attack categories, and in the final step, the resulting dataframes undergo z-score normalization for scaling purposes.

The second section of this chapter is dedicated to model selection, explaining the algorithms employed in this study for constructing the IDS. Initially, two baseline models are employed for the two distinct tasks: a logistic regression for the binary classification and a decision tree classifier for categorizing different types of attacks. The models in which the comparison between these two classification tasks is observed are two ensemble models, specifically the random forest and extreme gradient boosting, alongside two deep forward neural networks build with three dense layers and loss functions tailored to the respective classification tasks. Each model, both during its initial development as an IDS and subsequently with the different data modifications, undergoes evaluation based on accuracy, and additionally, considering their application in classification tasks, precision, recall, F1 score, and AUC metrics are employed. Together with these metrics, plots depicting confusion matrices and ROC curves are generated.

Within this study, two distinct methods are employed to simulate poisoning attacks on the dataset. The first method, referred to as "label flipping," involves shuffling a portion of the label column: this improper association of targets with each row results in incorrect algorithm training, and to assess and compare the consequences of this manipulation, increasingly higher fractions of the label columns are examined. The second method of data poisoning simulation acts on the predictors and employs a technique known as "feature modification", and in this study, this involves introducing random noise to ten variables within the dataset, with tests conducted at varying levels of noise intensity.

The evaluation measures obtained from the models across different dataset variations are analysed and compared in the “Results” chapter. With the clean dataset, the best results are achieved by the

neural network for both binary and multiclass tasks; also the ensemble models achieve acceptable accuracies, specifically, the second-best results are obtained by gradient boosting in the binary task and by random forest in the multiclass task, although gradient boosting outperforms random forest on the training set accuracy. Notably, the second classification task appears considerably more complex than the binary task, as indicated by the overall lower results, for instance, the decision tree performs inadequately, while logistic regression yields acceptable outcomes. The discrepancies in precision, recall, and F1 values between the two tasks highlight the challenges posed by class imbalance in the dataset, particularly for certain attack types, that are not identified by the models. Furthermore, the higher precision values relative to recall values suggest an elevated number of false negatives.

Subsequently, the results of models subjected to label flipping are analysed, employing different percentages of shuffled labels. This type of attack significantly impacts model accuracy, with performance degradation decreasing steadily as the proportion of mixed labels increases. Notably, neural networks are those heavily impacted by label flipping, performing less effectively than ensemble models when 90% of labels are flipped. The decrease in performance is also evident in precision, recall, and F1 values, especially in the multiclass task, while AUC remains relatively higher.

The second method employed for poisoning simulation, feature modification with random noise, shows a significantly lower impact on model performance: increasing the range of random values added to the features does not exhibit a consistent pattern of performance decrease, and the changes in metrics are very small compared to the previous attack method. Ensemble models appear to be the most affected, but the decrease in performance is also minimal for them. Specifically examining training accuracies reveals that the models have adapted well to these changes, showing minimal changes, with slightly higher variations in the test set. Other classification metrics also show minimal changes, with lower values observed in the same instances where accuracies declined.

In the chapter on preventive measures, various defence strategies against data poisoning are explored, with practical examples of their application. The foremost preventive measure in this context is data authentication, encompassing data integrity and authenticity verification, often obtained involving cryptographic techniques. Additional defences explored rely on data science methods and include outlier detection, data augmentation to enhance training dataset variability, adversarial training designed specifically to counter noisy data or data poisoning, and the use of a validation set, an additional split of the dataset instances used to enhance model training robustness; moreover, the

concept of human in the loop and digital twins are mentioned, to define a complete cybersecurity framework.

The main conclusions drawn from this study are summarized in the seventh chapter. It is evident that machine learning facilitates the creation of an efficient IDS system, with ensemble models and neural networks proving suitable for the task; however, distinguishing between different types of attacks is significantly more complex than just detecting suspicious traffic. It is then clear that data poisoning through label flipping has devastating effects on model performance, while feature modification, as applied in this study, has minimal impact. Exploring potential preventive measures underscores the strides made in cybersecurity against data poisoning, thanks to data science techniques, however, each defence strategy has its limitations, necessitating organizations to assess their risks and integrate multiple strategies into their defence systems.

2. Literature Review

There is a vast amount of technical literature in the computer science field on the technologies that are the subject of the analysis in this work. The aim of this section is to frame the state of the art of both IDS, with a focus on those built with machine learning and their usage in IoT, and data poisoning, with its major risks and potential safeguards.

2.1. Intrusion Detection Systems

Intrusion detection systems are a vital component in protecting communication networks due to their ability to monitor network traffic and spot any harmful or unauthorized activity within an IT environment. IDSs operate by examining all incoming and outgoing traffic to spot trends that might indicate malicious behaviour. They may accomplish this task by evaluating network traffic in comparison to a set of pre-established guidelines or standards that specify what kind of activity is normal or suspect. The system should then immediately notify the administrator of a potential intrusion whenever it detects any abnormal activity.

In a comprehensive review, Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin and Kuang-Yuan Tung (2013), provide an in-depth analysis of different IDSs available, presenting their features, architecture, and limitations. The aim of the authors was to provide a structured approach to analyse the IDSs, focusing on the methods used by the systems, that could be grouped in three areas: signature based, anomaly based, or specification based.

Given their purpose and operating way, with proper data collection, an IDS can be constructed through a classification task operated by machine learning algorithms. This approach was explored by Hongyu Liu and Bo Lang (2019), in a review of 115 papers on the application of deep learning and machine learning techniques for the synthesis of IDSs. The authors divided the publications into three groups, namely supervised learning, unsupervised learning, and deep learning, and examined the contributions made by each category to the field. The results shows that supervised learning is the most popular approach, while for unsupervised techniques there is less literature even if the authors identified it as a promising direction for further development; deep learning is recently gaining popularity due to their ability to handle large amount of complex data, even if computationally expensive. The dependency of these techniques on the quality and quantity of data for the specific situations is still an issue.

Another paper, with an approach similar to the previous one but considering more complex techniques, is the work by Dilara Gümüşbaş, Tulay Yıldırım, Angelo Genovese and Fabio Scotti (2020); the authors discuss three types of databases used in IDSs: Packet-level databases, which focus on capturing individual data packets exchanged in a network, Flow-level databases, focusing on collecting flow statistics, such as the number of packets and bytes exchanged between hosts, and Host-level databases, that capture data at the endpoint, including variables such as the records of system logs and event logs. The paper discusses several deep learning methods, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), that seem to be very effective in identifying network traffic anomalies, while Autoencoders (AEs) have great performance in anomaly detection with compressed data. The main issues highlighted in the paper are data quality, high feature dimensionality and models' interpretability.

Therefore, according to the literature, it is possible to create excellent detection systems with machine learning, while keeping in mind how crucial it is to choose the best algorithm and data pre-processing methods. These methods come with numerous applications and the literature provides analysis of some of these uses in detail.

The Internet of Things (IoT) has resulted in outstanding changes in the way we live, work, and communicate. However, with the increasing growth of connected devices comes an increase in security vulnerabilities. One of the most important aspects in protecting IoT security is detecting any malicious activity performed by possible intruders. Machine and deep learning-based intrusion detection systems are emerging as a potential remedy for IoT security challenges. Javed Asharf, Nour Moustafa, Hasnat Khurshid, Essam Debie, Waqas Haider and Abdul Wahab (2020) provide a systematic review of over 100 research papers, conference proceedings, and books published between 2010 and 2020. The results show the differences in three different approaches to build an IDS: anomaly-based, signature-based, and hybrid detectors. Some issues in this context are represented by limited computational resources and unreliable network connectivity, but the authors provide some hints to overcome them, and propose as further directions the integration of IoT data with cloud computing and the use of federated learning for distributed IDSs.

An interesting article focusing on a particular branch of IoT is that by Amar Amouri, Vishwa T. Alaparthy, and Salvatore D. Morgera (2020), that discusses the challenges of securing mobile devices, that have limited computational resources but are still, if not more, exposed to threats. A machine learning approach results suitable for this task, where the proposed system relies on a feature extraction module that collects data in an ordered way from the network traffic of the devices, to use then these data to classify traffic as normal or malicious.

One of the most recent publications in this field is by Andrea Pinto, Luis-Carlos Herrera, Yezid Donoso, and Jairo A. Gutierrez (2023), focusing on critical infrastructure, such as transportation systems or power grids, and the need for reliable and effective IDS to protect them from cyber-attacks. Here a summary of several machine learning-based IDS is provided, with a particular focus on ensemble models, with a complete review of their advantages and limitations. The authors highlight some of the major issues in these field, such as the lack of labelled datasets, data imbalances, and the need for real-time processing; they also stress how crucial it is to incorporate domain-specific expertise while creating these systems in order to increase their efficacy and accuracy.

Another type of technology that is worth mentioning when discussing IDS are digital twins, virtual representations of a real-world infrastructure designed to reflect its behaviour, performance and characteristics in a highly detailed and dynamic manner. These systems are increasingly being implemented as fundamental cybersecurity tools, as they offer great advantages in managing the risk environment of complex systems, as explored by Andrea Salvi, Paolo Spagnoletti and Nadia Saad Noori (2021), proposing a model to provide cyber critical infrastructure's actors with situational awareness, a common understanding of incidents and a greater ability to respond. Other industries, with the advent of digitalization, are also using AI based digital twins as their main defence measure, integrating IDS systems into these models, as shown in the industrial control system framework proposed by Seba Anna Varghese et al. (2022)

According to many of these publications, researchers face an issue in finding good data to work with in order to create an effective IDS: for this reason, there exist some public collections that can be used for research purposes, and one of the most comprehensive and popular, the UNSB-NB15 dataset, will be used in this work. The UNSW-NB15 dataset is a network-based intrusion detection system (NIDS) dataset consisting in a collection of network traffic that includes both normal and malicious activity, and it was developed by The University of New South Wales in Sydney. The dataset contains 2.5 million network flows and 49 network attributes (such as source and destination IP addresses, port numbers, packet size and so on). The malicious traffic in the dataset contains a variety of attacks, and each observation is labelled as normal or as the attack type it indicates, allowing for a supervised learning approach. The complete description of its features and the process to create the dataset can be found in the paper *"UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)"* By Nour Moustafa and Jill Slay (2015).

The UNSW-NB15 dataset is an important resource for developing and testing NIDS algorithms and approaches, and there are consequently many articles showing works on it. The authors of the dataset themselves, Moustafa, Nour, and Jill Slay (2016), present an evaluation of IDS built on the NB15 in

comparison with a previous dataset, the KDD99, showing that it is more updated in term of attack types, but also more challenging. The authors in subsequent years continued to use the dataset in innovative research methods related to NIDS, explaining in some of these works further details of the dataset and presenting interesting application such as: *"Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." (Moustafa, Nour, et al., 2017)*, *"Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." (Moustafa, Nour, et al., 2017)*, *"NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems." (Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann, 2020)*.

Other studies show how the dataset is suitable for analysing IoT contexts. The paper again by Nour Moustafa, et al (2018) adapt the NB15 to this purpose by proposing a new set of statistical flow features that capture the unique characteristics of IoT traffic. The results show that the proposed ensemble model as a combination of some popular machine learning techniques (namely decision trees, neural networks, and support vector machines), trained with those data outperforms other state of art IDSs. Another specific issue investigated through these data, for which was necessary to manage challenges such as the heterogeneity of IoT devices and the lack of standardization in their protocols, is presented in the paper by Nickolaos Koroniotis, Nour Moustafa et al. (2017). Here, again, classification via machine learning models, along with suitable pre-processing and feature extraction, results effective, detecting botnet activities in the network traffic, and the study accomplish to provide a framework for this scenario of network forensic.

2.2. Data Poisoning

Data poisoning poses a growing challenge for machine learning and artificial intelligence systems. This kind of attack happens when malicious actors deliberately inject false or harmful data into a system, usually with the purpose of altering the results. Data poisoning can lead to incorrect conclusions, making the models useless, and in worst cases even to system failures. In whatever context the machine learning system operates, if carried out in such a way that corrupted data are not easily distinguishable from others and are thus taken into account in model training, data poisoning can be a huge threat: harmful decisions (if unfiltered) are not a potential danger only to researchers or companies, but to anyone. For example, an AI-powered financial trading system could be fooled to purchase stocks at inflated prices, or an AI-powered medical diagnostic system could be duped into producing incorrect diagnoses.

The literature on this type of attack is extensive, and one of the first works to provide a comprehensive framework, focused on one specific machine learning model, is that by Battista Biggio, Blaine Nelson, Pavel Laskov (2012), in which they discuss the different kinds of poisoning attacks that can be conducted against Support Vector Machines (SVMs), not only data poisoning, but also model poisoning, concerned on modifying the parameters of the algorithm, and their exploitation as backdoor assaults. The article describes how an attacker might alter training data by adding or removing specific features, changing the labels of instances, or constructing synthetic examples; it also goes into the various methods an attacker could deploy SVMs to fool the classifier during the testing phase. This work proposes some safeguard measures, among the strategies discussed are outlier detection and adversarial training. The authors investigate the advantages and disadvantages of each strategy and formulate some recommendations for their application in real world circumstances.

In the past decade, poisoning attacks have been a popular topic in research, and a recent and extensive overview is provided in the paper by Miguel A. Ramirez et al. (2021). This comprehensive review of the existing literature includes many attack scenarios, on different type of models and performed with different techniques, analysing their effect on classifiers. The survey takes in account also the defence mechanisms, showing that there exists many countermeasures based on different methods, among which it is possible to find robust training, input verification, outlier detection, model-based defences, and detection-based defences; however, the authors conclude by highlighting the need to develop more robust and effective defences, since the existing ones are not always effective and this kind of attacks are still a big threat.

As introduced in the previous section, IoT networks connect many devices through the internet, allowing them to share each other huge quantities of data; this type of networks imply multiple vulnerable points that have access to data, often sensitive, and for this reason data poisoning represents a major threat for IoT. Poisoning attacks can occur in several ways, ranging from changing sensors' records, adjusting control signals, or inserting fake data into the IoT system, and as outcome, the attack can compromise the accuracy and reliability of the information on which systems rely on, leading to wrong or harmful decisions. An interesting paper that explores, among other threats, data poisoning attacks on IoT is that by Murat Kuzlu, Corinne Fair, and Ozgur Guler (2020). This work discusses many cyberattacks that could be conducted on these networks and provides insights on how AI can help combat them by providing real-time threat analysis and identification of zero-day attacks. Besides illustrating the potential of AI to create more secure IoT ecosystems in the future, it explains

how AI models themselves could be used by attackers, and one of the attacks that can be performed in this manner is precisely data poisoning or false data injection.

Another very informative paper that relates to the IoT field is by Gan Sun, Yang Cong, Jiahua Dong, Qiang Wangand Ji Liu (2015), exploring the context of federated machine learning, that entails training machine learning models with data from different sources without explicitly sharing them. This approach is especially effective in businesses where data privacy is critical, such as healthcare. Despite its advantages, federated machine learning is vulnerable to data poisoning attacks, in which malicious parties attempt to feed corrupted data into the system in order to compromise its integrity. The authors claim that these assaults can be more severe since the dispersed nature of federated learning makes detection and mitigation more difficult. To demonstrate the consequences of data poisoning attacks, after explaining different types of them, the authors take the example of a malicious actor using a target attack to change the model's prediction for a specific input, that is misclassifying a cancer diagnosis in healthcare.

The last publication in this literature review, which is still about data poisoning in the IoT environment, has among its authors the creator of the UNSW-NB15 dataset, and part of the work is carried out with these data. The aim of Corey Dunn, Nour Moustafa and Benjamin Turnbull (2020), is to present a framework for integrating classical machine learning approaches with anomaly detection and more complex deep learning algorithms, resulting in more robust and long-lasting machine learning models that can withstand malicious attacks. The authors evaluate then the effectiveness of this framework by conducting several experiments and use a real world IoT dataset to demonstrate its applicability; their results show an enhancement in the resilience of models to data poisoning, along with mitigation of the possible collateral damages.

3. Research Questions

This chapter defines the questions that this thesis will attempt to answer, in order to trace the progress of the analysis that will be carried out in this work and clarify the main aims.

The initial aim of this work is to demonstrate how effective machine learning algorithms and neural networks are for developing an IDS by classifying internet traffic data as pertaining to normal or malicious activities. Along with the first stages of data selection and pre-processing, the use of appropriate models will be critical in ensuring the task's success. In machine learning, the model selection stage has a substantial impact on the results: the choice of an algorithm for a task determines the prediction accuracy, the ability to interpret and explain outcomes, and the efficiency in dealing with big datasets or real-time applications. Furthermore, each model has its own characteristics that makes them suitable for specific tasks, and other aspects such as the system's robustness against noise and outliers, generalization capabilities to previously unseen data, and compatibility with certain data features are all affected by model selection. Model selection is even more critical when developing an Intrusion Detection System: the IDSs' domain requires models capable of processing and analysing massive volumes of network traffic data in real-time while reliably discriminating between regular and malicious activity. It is critical to use Machine Learning and Deep Learning models that have high classification capabilities, solid feature representation, and the capacity to handle dynamic and developing attack patterns.

In this work, two different classification tasks will be carried out: The first is a binary classification, therefore a task that involves classifying instances into one of two exclusive categories, in which the model's output typically represents the probability or confidence of belonging to one of the two classes; here, the goal is to predict the binary outcome of "normal activity" or "attack activity", encoded in 0 or 1 values. In the dataset used for this work, there is also a more specific label column, specifying for the attack instances which kind of cyber threat is at stake; therefore, the second task is a multiclass classification, meaning that it involves classifying instances into one of many mutually exclusive categories, so to assign each single instance to the most appropriate class out of the normal activity or the possible cyber-attacks.

The first part of the work is therefore concerned with the training and fitting of four models, with the goal of performing the two aforementioned classification tasks on the dataset. The algorithms used and the measures to evaluate their performance will be explained in the following chapter, and the research question to which the analysis tries to answer can be summarized as follows:

How good are the performances of machine learning and deep learning models in an IDS classification task? How much does their performance change between just detecting suspicious activity or recognising the type of attack in progress?

As anticipated in the previous chapters, the second main aim of this work is to simulate a data poisoning attack on the previously trained algorithms and evaluate its effects. Data poisoning attacks attempt to modify training data in such a way that the integrity and performance of machine learning models are compromised; the primary goal of this kind of attack is to make the models worthless, compromising their performance and accuracy; a model compromised in this way will no longer be able to obtain good results for its purpose, making it unreliable and harmful, as it might lead to incorrect conclusions.

Evaluating the effects of a specific kind of data poisoning attack is fundamental to have an idea of which and how serious the vulnerabilities of a machine learning system are, especially in a world where these are increasingly being used in particularly sensitive contexts, such as computer security in this examined scenario. Knowing the magnitude of a risk is always at the core of the process that leads to the development of appropriate defences against it. By building the intrusion detection system with different methods, it will be interesting to see how proportionally sensitive these algorithms are to a certain attack, always evaluating this with the proper measures.

By means of data poisoning simulations with two different poisoning methods, both of which have been tested under increasingly difficult circumstances by leveraging some parameters that aim to incrementally decrease the effectiveness of the classifiers, it will be possible to precisely monitor their effect using the evaluation measures. The second key goal of this work can thus be summed up in the question:

What is the impact of a simple data poisoning simulation on these models? How much do their performances change according to the different methods and the several intensity levels of poisoning?

This work, and especially its practical component, intends to demonstrate how data science approaches is applicable in the field of cybersecurity, while also being capable to produce harmful situations. Once the results will be obtained, it will be necessary to investigate how, once again using data analysis techniques or manipulations during algorithm training, the danger generated by a potential data poisoning attack might be reduced. In the final section of the thesis, some of the most typical preventive countermeasures, employed to ensure the proper operation of an intrusion detection system, and related with the works examined in the literature review will be discussed. The last

objective of this work is therefore to understand the functioning of these possible defences, and to frame their usage is some possible real-life scenarios, as summarised in the question:

What preventive measures against data poisoning can be implemented in the construction of this type of IDS? What are their advantages and limitations?

4. Methods

The code required to carry out this study will be reviewed in detail in this chapter, beginning with the dataset and progressing through the algorithms employed, as well as the assessment measures and data poisoning approaches. The technical aspect was carried out entirely on Python notebooks, which can be found in this GitHub folder:

https://github.com/DavideRosatelli/Intrusion-Detection-System-Data-Poisoning/tree/main

Much of the technical work is focused on the training and testing of machine learning models and neural networks, for which Google Colaboratory Pro was utilized, which sped up the process by using a TPU and extensive RAM.

4.1. Dataset Description and Preprocessing

The UNSW-NB15 is a large cybersecurity dataset, created by researchers at the University of New South Wales (UNSW) in Australia to aid in the evaluation of IDSs and anomaly detection methods. Its complete description can be found in the paper *"UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)" (Moustafa, Nour, and Jill Slay, 2015).* The dataset was produced in a controlled environment by simulating realistic network traffic and by integrating normal network operations and numerous simulated attacks. To generate this traffic, the researchers employed the IXIA PerfectStorm tool, which faithfully simulates real-world network activity, and then used multiple network assaults to generate instances of malicious traffic. The dataset is the outcome of capturing and recording this mixed network traffic, which provides a diversified and complex set of network flows.

The dataset contains a diverse variety of features derived from network packets and logs, that can be categorized as follows:

- Basic features: These features capture fundamental information about network traffic, such as source IP address, destination IP address, source port number, destination port number, protocol type (TCP, UDP, ICMP), packet length (in bytes), and timestamp.
- Content-related features: This category includes features related to the content of network packets, such as payload data in hexadecimal format, flow duration, packet count, and byte count in a flow.

- Traffic features: These features focus on characteristics related to network flows, such as the number of packets or bytes in forward and backward direction, and the number of specific flag packets (e.g., FIN, SYN, RST, PSH).
- Labels: There are two label columns in the dataset, making it suitable for supervised learning. Each instance is classified as 'Normal' or 'Attack' in the first label column, while in the second column the attack type is specified, and the dataset contains nine different cyberthreats.

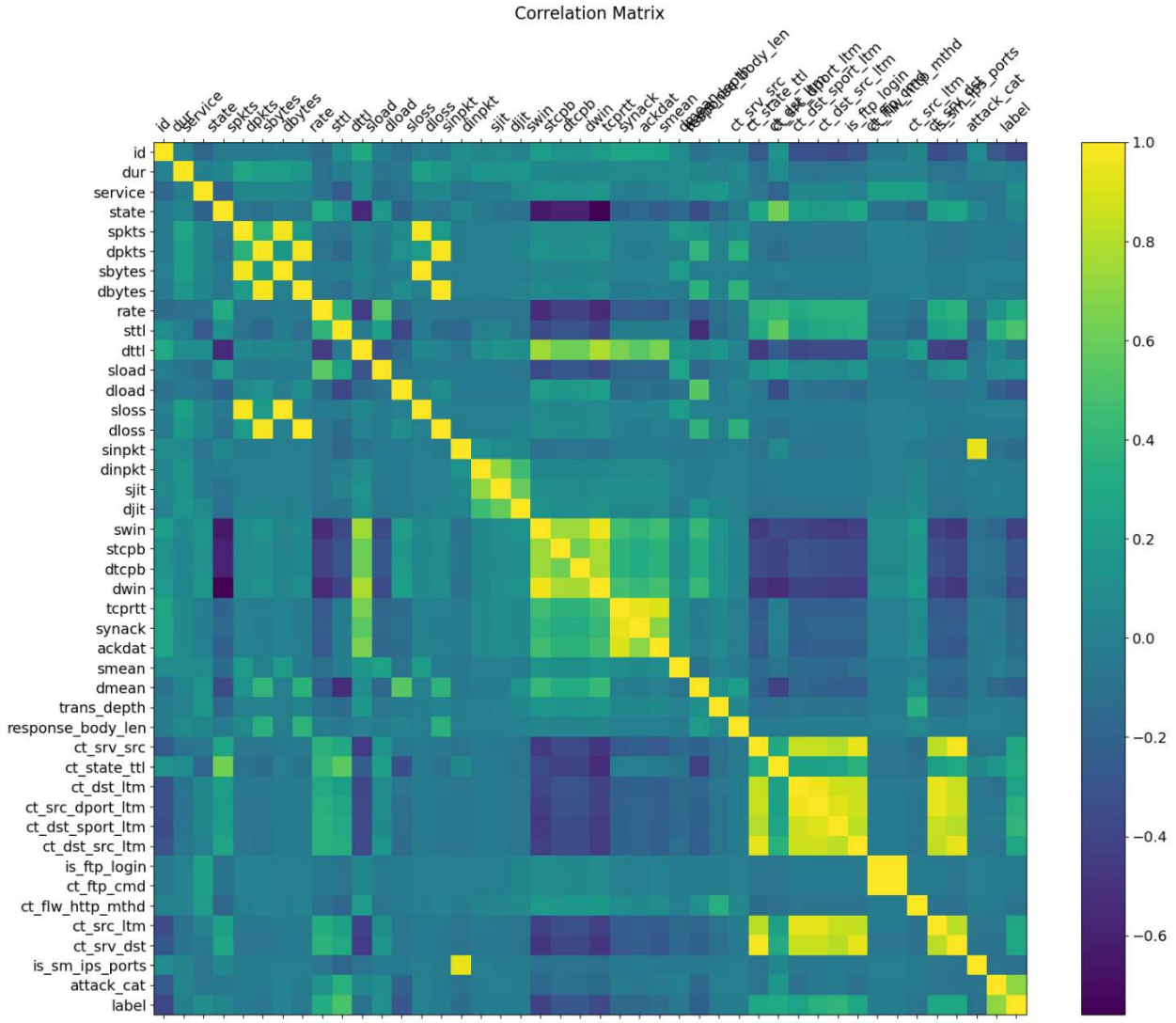
The UNSW-NB15 dataset contains a total of 2,540,044 instances, stored in several csv files that are available to download for research purposes; the full dataset is very big, and for this reason the authors provide also a subset of the full dataset already split into train and test samples, that are big enough to fruitfully train machine learning algorithms, and in this work those two files are used.

The dataset is therefore very well structured, and no great effort was required for the preprocessing part. There are no missing values in the dataset, but it was necessary to transform all instances into finite values, as some had values in a form unsuitable for algorithms. The libraries used to manipulate the data in this code section were Pandas and NumPy.

Almost all the variables are in numerical form (either integer or floating-point), except for the columns *proto*, *state*, *service* and the second label *attack_cat*; the former, indicating the protocol, has been eliminated, while all the others have been mapped to numbers using an ordinal encoding, via Python dictionaries.

Another crucial step in the preprocessing phase is to determine the most significant variables for the task at hand, and a correlation matrix is a straightforward way to do this. Correlation is a statistical metric that expresses how closely two variables are associated, and so how much when one changes, the other is statistically likely to change, even if there is no direct cause-effect relationship. The variables most correlated to the target are generally the most important for machine learning models, and to identify them, the correlation matrix is represented by a heatmap in this work, where the names of the variables are on rows and columns, and the cell corresponding to their intersection is their correlation value, indicated by a dark colour tending to blue if low, and a brighter colour tending to yellow if high. The result, easily achievable with Python through the library Pandas, is shown in *Figure 2*. Identifying the factors that are most correlated with the target will be also critical for a specific type of poisoning simulation, as shown in the following sections.

Figure 2: Correlation Matrix



Analysing the dataset, specifically the two train and test files provided by the authors, it is clear that these are not optimally balanced, most likely because the test is intended to be used as a final test after training the algorithms on the entire dataset: in fact, this contains mostly instances classified as attacks, and the number of observations is higher than in the train, which appears instead to be a random sample of the entire dataset. To avoid over- or underfitting due to this unusual distribution, the two dataframes were merged and then separated again in a train (70% of the observations were chosen at random, for a total of 172630 rows) and a test set (30% of the observations were chosen at random, for a total of 85028 rows), with labels' stratification. As a result, the two target columns were separated, yielding a total of six different dataframes that were stored externally as csv as follows:

X_train, X_test, y_train, y_test, y_multi_train, y_multi_test

The “X” family of dataframes contain all the decision variables, while the first two “y”s dataframes will be used for the binary classification of ‘Normal’ and ‘Attack’, and the “y multi” dataframes will be used for the multiclass classification in which also the attack category should be identified. To be aware of the distribution of the classes, especially in the test set in order to have the values to compare with the confusion matrices of the models, that will be explained later, using a simple function the number of occurrences for value in the set *y_test* and *y_multi_test* were printed, showing that there are 30685 instances of normal traffic and 54343 instances of attack classes, divided in:

- 19493 Generic
- 14798 Exploits
- 7903 Fuzzers
- 5312 DoS
- 4653 Reconnaissance
- 850 Analysis
- 771 Backdoor
- 509 Shellcode
- 54 Worms

In order to get a general idea of the values’ distribution in the variables of the dataset, they were represented one by one by line plots. These graphs, given the big number of values, are not very informative, but they are enough to note that the distributions of the features are very different. Furthermore, it is useful to standardize the magnitude of the values represented in the features as many statistical learning classifiers are sensitive to different features having different scales, yielding an implicit weighting phenomena. Feature scaling will also be important to carry one of the poisoning strategies described in the next sections. For these reasons, the final stage of preprocessing is to normalize the dataset using the *StandardScaler* provided by the Scikit-Learn library. Standardization, also known as *z*-score normalization, is a critical preprocessing step, especially when working with algorithms that are sensitive to feature scale, such as gradient descent or distance-based algorithms.

The *StandardScaler* first calculates the mean (μ) and standard deviation (σ) of each feature independently; then, each feature is standardized by subtracting the mean and then dividing by the standard deviation. The formula for standardization is given by:

$$z = (x - \mu) / \sigma$$

where x is the original feature value, z is its standardized value, μ and σ are the aforementioned mean and standard deviation of the feature itself. The *StandardScaler* returns a modified version of the

dataset after applying the formula to each feature, yielding features with zero mean and unit variance. After this step, the line plots of all the features show generally a better distribution of values, as can be seen in Appendix A.

4.2. Machine Learning Models and Neural Network

In order to build the intrusion detection system, and subsequently put it to the test with the data modified by the various poisoning techniques, several algorithms were used, which will be explained in this section, starting with the simplest and most interpretable, and proceeding with the most complex ones.

The model used as baseline for the binary classification task is a logistic regression, a regression analysis that is specifically designed for predicting the probability of an instance belonging to one of two possible classes. Mathematically, a sigmoid or logistic function is used by the model, that is:

$$f(z) = \frac{1}{1+e^{-z}}$$

$$\text{where } z = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$$

This z is a linear combination of the input features and their associated weights, plus the bias term ω_0; the output of the function represents the estimated probability of an instance to belong to positive class, and it is therefore assigned to it if above the 0.5 threshold. The logistic regression is widely used due to its interpretability and leads to good accuracy when the relationship between the features and the target variable is not overly complex; this type of model can easily be implemented in Python via the Scikit-Learn library.

A logistic regression is the typical baseline model for a classification task, but it is only suitable for binary outputs; a simple algorithm that is instead suitable for any type of supervised output, and will therefore be considered as the baseline for the multiclass task in this work, is the decision tree, which in this work will be implemented using the *DecisionTreeClassifier* class of Scikit-Learn to categorise the different cyber-attacks in the dataset.

A decision tree classifier can be imagined as a flow chart starting from the root node and progressing down the leaf nodes, in which each internal point entails a decision on which path to follow, based on one of the features' values. The algorithm chooses the factor that best separates the data into various classes at a given node, with the goal of selecting features' values that result in the best feasible separation of classes at each step, which is accomplished using loss methods such as

information gain and Gini impurity. Building a decision tree entails iteratively splitting data into subgroups, the splitting process continues until a requirement is met, in the code of this work the trees were set with a maximum depth of 10. Decision trees are widely used models due to several advantages, such us their interpretability and the ability to handle both categorical and continuous features; however, they are prone to overfitting and not suitable for complex tasks, and to cope with these limitations, alternative machine learning algorithms known as ensemble tree methods were developed, of which the two algorithms discussed below are part. Ensemble tree methods are powerful machine learning techniques that combine multiple individual decision trees to create a more robust and accurate predictive model, widely used for both classification and regression tasks. In this work, for both the binary and the multiclass task, random forests and extreme gradient boosting will be used.

A random forest is an ensemble method that builds several decision trees during its training, to aggregate then their output and make the final prediction. Random forest employs a technique known as bootstrapping, that entails generating many random subsets of the training data with replacement, to train with each of them an individual decision tree. At the end of the training, the class that receives the most votes from the individual trees is the final prediction, as showed in *Figure 3*. Furthermore, a subset of features is chosen at random from the whole set at each split point of the decision trees, to prevent a single variable from dominating the decision-making process and encourage each tree to evaluate other features. In Python, this work used the *RandomForestClassifier* provided in the package Scikit-Learn, aggregating the results of 30 decision trees each with a maximum depth of 10. Random forests provide several advantages, such as robustness to overfitting and good handling of

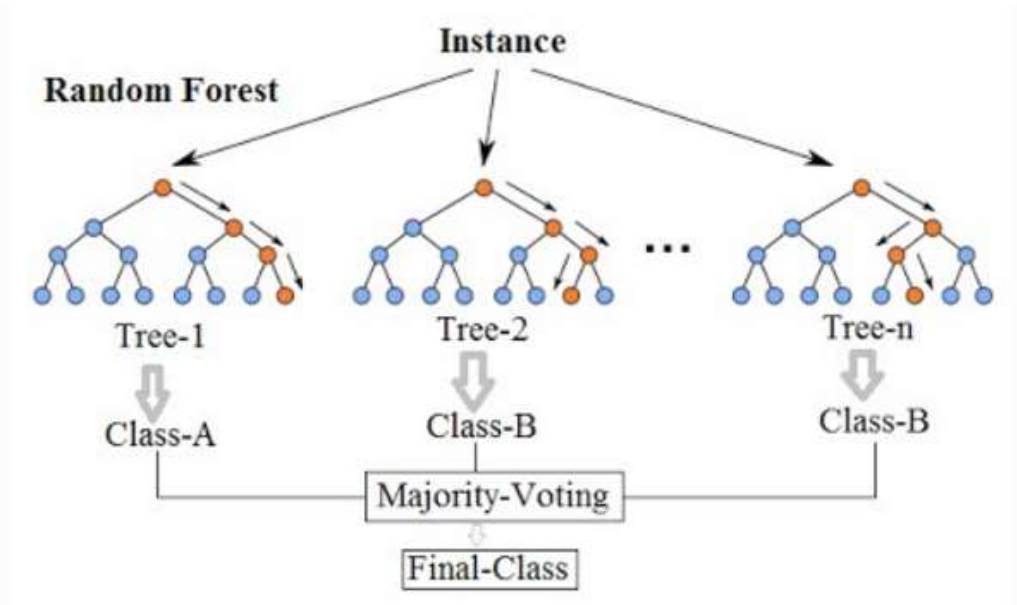


Figure 3: Random forest process scheme

noisy data, and it will be therefore interesting to compare its results with another popular ensemble method.

The XGBoost or extreme gradient boosting is an advanced ensemble learning method, enhancing the gradient boosting algorithm, with the main idea of building several decision trees sequentially, where each new tree adjusts the weights provided by its predecessor, with the aim of optimizing an objective function. A simplified architecture of this model is showed in *Figure 4*.

To minimize the objective function, XGBoost employs gradient descent optimization: it computes the gradients of the loss function with respect to the model's predictions and modifies the model's parameters (trees' weights) to minimize the loss. Differently from the random forest, the XGBoost considers all features at each split in the trees, assigning different weights for them based on their importance, and the final predictions are obtained by updating the contributions of all the individual trees. The package xgboost allows users to choose the objective function in its Python implementation, and in this study, the logistic function was used for the binary task, while a softprob function, a modified version of the softmax, which will be further explained later, was used for the multiclass task. This model is popular due to its ability to generally get high accuracy, and its flexibility in loss functions, but respect to the algorithms seen above, is less interpretable.

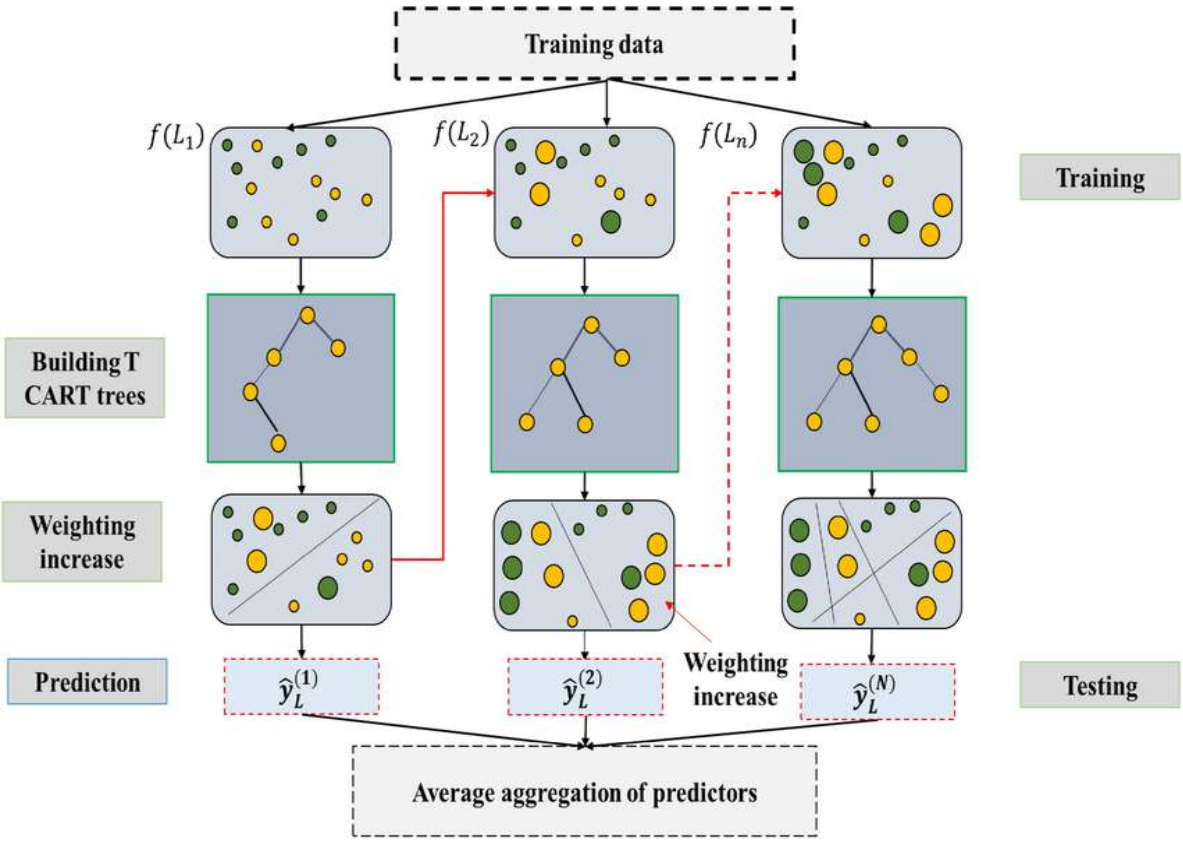


Figure 4: XGBoost process scheme

A feedforward neural network is the final type of algorithm employed in this work, and it was used for both the binary and multiclass tasks. A feedforward network, also known as a dense layer neural network or a multi-layer perceptron (MLP), is a fundamental structure in deep learning. It is made up of numerous layers of interconnected neurons (nodes) structured in a sequence, with each neuron in a layer is connected to all neurons in the preceding and following layers, as showed in *Figure 5*.

In Python, neural networks are easily implemented through the Keras library, which makes it possible to order layers of different types and specify numerous hyperparameters. For both tasks, the networks are composed of 4 dense layers: the first two layers composed by 128 neurons each, and the next two by 64 neurons each, while the output layer will have size 1 for the binary case and 10 (i.e., the number of classes) in the multiclass case. Each dense layer needs an activation function, which for the input and hidden layers is in this case a rectified linear unit (ReLU), widely used because it is simple but effective due to its property of introducing non linearity in the relationships between the data; as far as the outputs are concerned, first a sigmoid activation will be used, the logistic function that maps input values to the range (0, 1), commonly used for binary tasks, while for the multiclass task a softmax will be used, an extension of the sigmoid function that generalises to multiple classes, that is mathematically:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

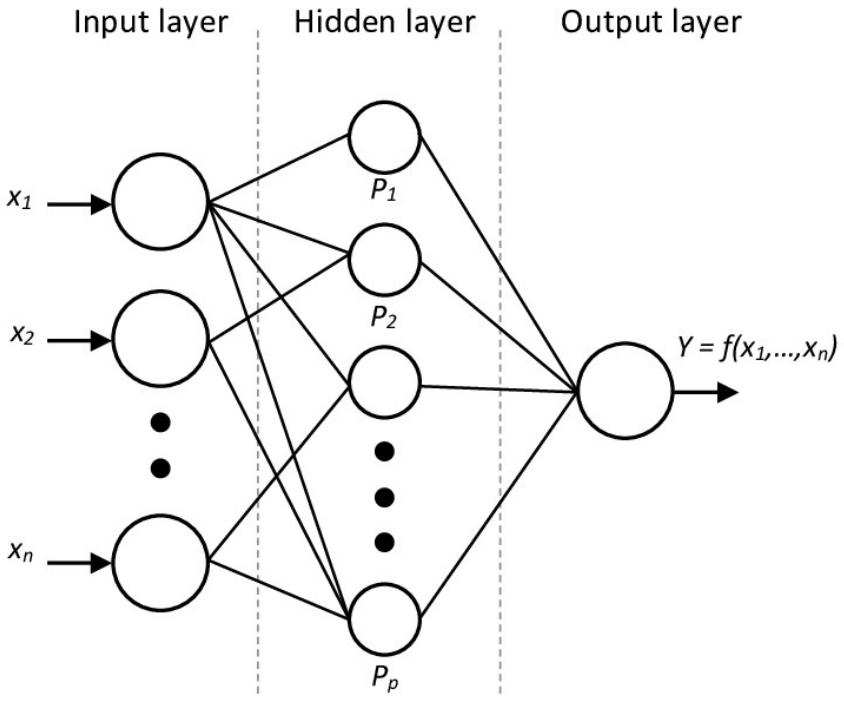


Figure 5: Feedforward neural network process scheme

where s_j are the scores inferred by the network for each class in C.

The loss function and the optimiser are the other two key parameters that must be set in a neural network. The loss function assesses the difference between the predicted outputs and the actual labels; it must thus be minimized and tailored to the type of work; for instance, in the first task, it is used a binary cross-entropy, while in the second, it is a categorical cross-entropy, that mathematically read as:

Binary Cross-Entropy ➔ $$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1)$$

Categorical Cross-Entropy ➔ $$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

where t_i and s_i are the probability score for each class in C.

The optimizer, on the other hand, is the algorithm that updates the network weights to minimize the loss function, and in both tasks, the Adam optimizer, short for "Adaptive Moment Estimation" is used, which is popular due to its fast convergence, robustness to noisy gradients, and relatively low memory requirements. All the neural networks in this work are trained with a batch size of 128, and for 100 epochs, with a call-back required to stop the model when the loss increase.

Given the non-linearity of the dataset, comprehending a big ecosystem of features, the classifications required in this work will surely be more accurate when performed by algorithms with a high level of complexity. For this reason, the two models described at the beginning of this section, namely the logistic regression and the decision tree, will be used as baseline to observe the behaviour of a simple algorithm in the binary and multiclass task, while the main focus of the analysis, entailing the comparison of the results in both the tasks, are the two ensemble models and the feedforward neural network.

4.3. Evaluation Techniques

Once the different machine learning models have been trained in the various data configurations that will be used throughout this work, it will be crucial to evaluate their performance. In order to have a comprehensive understanding of the model's performance and its strengths and weaknesses in different aspects of the classification tasks, various evaluation measures will be used; those measures are defined and explained in this section and will be used to compare the computational results, and

thanks to these comparisons, conclusions can finally be drawn, trying to answer the research questions.

In machine learning, the basic, fundamental, and widely used evaluation metric is accuracy, measuring how well a specific model performs in its task, so, if as in this case the task is a classification, it measures how much a model correctly predicts the class labels of the input data points. Formally, accuracy is the ratio of the number of correct predictions made by the model to the total number of predictions it has made. For instance, if in a dataset of 1000 observations the model can correctly predict 800 labels, it will have an accuracy of 0.8, or 80%, so the higher the accuracy, the better is the model's performance.

In a classification task another fundamental tool to evaluate the models, is the confusion matrix, a table with combinations of predicted versus actual values. In a binary classification scenario, the confusion matrix is a 2x2 table, organized as can be seen in *Table 1*, while in multi-class classification, it has more rows and columns, depending on the number of classes. The values inside the binary confusion matrix are interpreted as follows:

- True Positive (TP) represents the number of positive instances correctly identified as positive;
- False Positive (FP or Type 1 Errors) represents the number of negative instances incorrectly identified as positive;
- True Negative (TN) represents the number of negative instances correctly identified as negative;
- False Negative (FN or Type 2 Errors) represents the number of positive instances incorrectly identified as negative.

Table 1: Confusion matrix scheme for a binary task

	Predicted positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

The values in the confusion matrix are the baseline to compute many others evaluation metrics: accuracy itself, in a classification task, can be mathematically defined as:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

As previously stated, accuracy is almost always used in model evaluation, but it is not always sufficient to understand how effective a model truly is at its classification task, and this work is exactly one of those cases: if the dataset is not well balanced, meaning that there are many more instances of one class than the others, the model may only be able to classify those correctly and still have good accuracy. In a situation involving a dataset as the one used in this work, if for instance 90% of the observations were normal Internet traffic and just 10% were attacks, an IDS could fail to recognize any attacks while still having a 90% accuracy. When the dataset is unbalanced, different metrics must be used to provide a clearer picture of performance.

Precision, recall, and F1 score are important evaluation metrics used in classification tasks. They provide valuable insights into a model's performance, especially when dealing with imbalanced datasets, and they can be derived from the values in the confusion matrix; their definition is derived from a binary classification context, but they could be applied also in a multiclass context.

Precision, also known as positive predictive value, assesses the model's accuracy in making positive predictions, estimating the proportion of true positive predictions out of all positive instances predicted by the model. A high precision score suggests that the model has a low false positive rate, which means it predicts fewer wrong positives, so the closer it is to 1 the better the model works. Precision is calculated mathematically as:

$$Precision = TP / (TP + FP)$$

Recall, also known as sensitivity or true positive rate, measures the capacity of the model to properly identify positive observations from the total number of actual positive instances in the dataset; again, the closer this value is to 1 the better the model works. Calculating the proportion of true positive predictions among all positive instances, it is determined mathematically like follows:

$$Recall = TP / (TP + FN)$$

The F1 score is the harmonic mean between precision and recall, mathematically calculated as:

$$F1\ Score = 2 * (Precision * Recall) / (Precision + Recall)$$

The F1 score has a maximum value of 1 (with precision and recall both perfect) and a minimum value of 0. It is especially useful when there is a trade-off between precision and recall. For example, a model classifying most of the observations as positive while have a lower precision, due to the number of false positives, but an high recall, and the opposite can also happen with high precision and low recall indicating more false negatives. As a result, the F1 score represents a more balanced measure of model performance.

The last evaluation method used along the analysis, is still derived from the values in the confusion matrix, and it is a graphical representation of a classification model's performance called "Area Under the Receiver Operating Characteristics Curve", or AUC-ROC curve. The ROC curve plots the relationship between the true positive rate and the false positive rate as the classification threshold varies, meaning that the values on its axis are:

$$y: TPR\ (recall) = TP / (TP + FN)$$

$$x: FPR = FP / (FP + TN)$$

The resulting curve can be graphically interpreted by stating that the nearest the elbow is at the top left corner, the better is the models performing; the scalar value resulting from the graph is instead the AUC, so the area under the curve, spanning from 0.5 (a random guessing) to 1 (a model with perfect discrimination ability).

Throughout the Python code, all those evaluation metrics were provided by the library Scikit-Learn, using the function *classification_report* included in its metrics. By means of a few adjustments to the data objects and the use of other libraries such as itertools and yellowbrick, all of these have been calculated for each model and neural network. In the cases with more classes to identify, the measures of precision, recall, F1 and AUC are computed for each of the different classes, and the same applies to the confusion matrix which will have dimensions 10x10, as many as the total number of categories.

4.4. Data Poisoning Techniques

This section describes in detail the approaches used to simulate data poisoning, which, as said, is an attack in which the attacker attempts to introduce properly designed malicious samples into the training dataset in order to affect the model's behaviour.

Two very distinct strategies will be employed in this work, both of which are simple but effective in demonstrating the influence of a data modification on model training. What will be done in both cases, in order to replicate a hypothetical real-life situation, is to make slight changes on the training set used to train the various models, so that their prediction accuracies decrease, leading to even less successful predictions on the test set. The key difference between the two methods is that the first will be applied to the labels of the training set, while the second will change the values of the features; both will be performed with multiple levels of intensity in order to monitor their effects by looking at the algorithms' evaluation measures.

The first method to obtain a simulation of data poisoning is known as label flipping, which involves as main idea to change the order of the target column: as a result, the model will no longer have confident references, as the instances in the training set will no longer be associated with their true label, but with a different one. The ability to find suitable parameters that associate a given label with specific predictor values is what allows machine learning models to produce accurate predictions and classifications, therefore, associating improper targets with each row will lead to an incorrect training of the algorithm. In the real world, to carry out a cyber-attack of this nature, one must be aware of the structure of the dataset, specifically the target column, and have the ability to modify it separately from the predictors; modifying the target is one of the most direct, simple, and effective ways to compromise a machine learning system, and the alterations can be even more specific, in order to train the classifier to make the most incorrect decisions.

In this work, the label flipping is accomplished in Python by simply shuffling a particular percentage of the label column. To examine and compare the consequences of this alteration, increasingly compromised dataset fractions will be examined, and all models will be retrained with the following percentages of randomly shuffled *y_train:* 15%, 30%, 45%, 60%, 75%, and 90%. The shuffling is obviously conducted on both the binary and multiclass label columns in order to run the analysis on all the previously descripted classifiers.

The second technique investigated in this work to replicate data poisoning is known as feature modification, since the values of the predictors in the training set are changed rather than the labels. In general, this type of assault is less direct than the one described above, but it is arguably more subtle, and if executed correctly, it is both difficult to detect and highly successful at decreasing the model performance. An attacker who wishes to undertake feature modification must be aware of the structure of the dataset and possess the ability to identify what to modify in order to make these alterations difficult to detect while being effective. When working on predictors, the changes that can be made are numerous, and inserting specific values into some or all the variables can force the model to predict exactly what is desired; to reduce the overall performance of a classifier, however, changing the covariates' values in a random manner should be sufficient.

In this study, the ten most correlated variables to the target columns were chosen by looking at the correlation matrix (*Figure 2*), namely:

service, sbytes, rate, sttl, sload, ct_state_ttl, ct_src_dport_itm, ct_dst_sport_ltm, ct_src_ltm, ct_srv_dst.

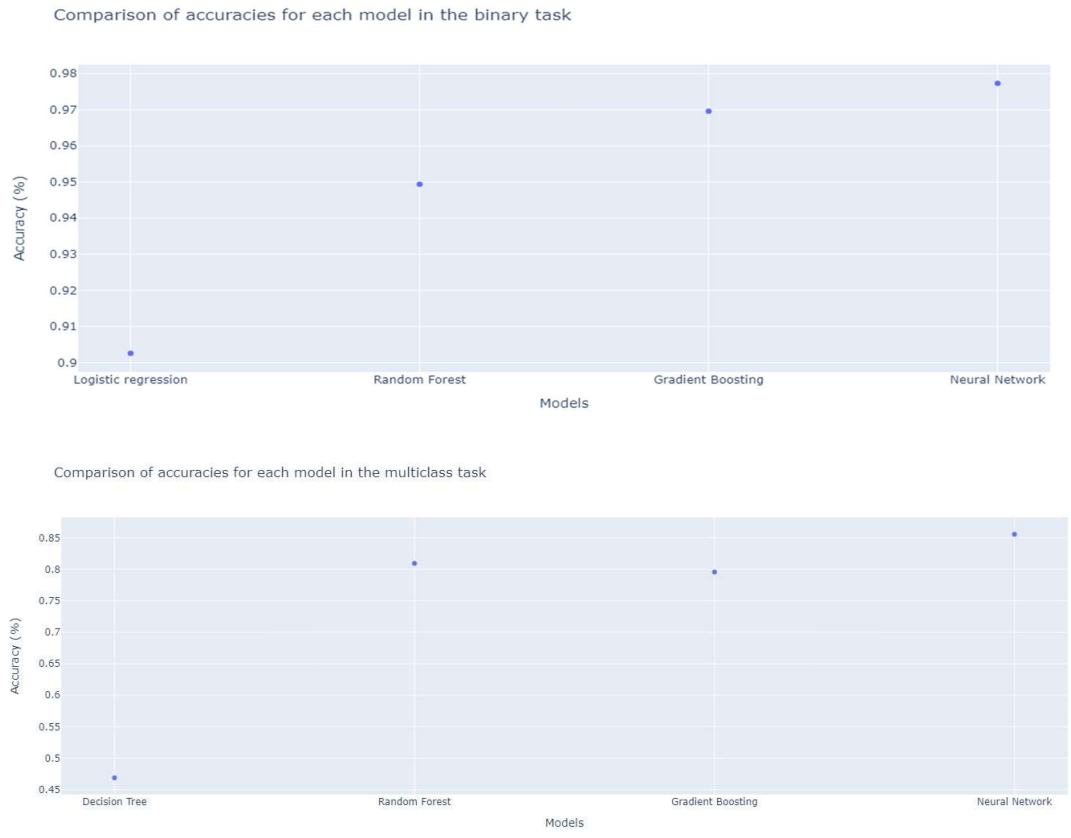
Then, to all the instances of these variables, a certain level of noise is added, meaning that their original value is summed with a random number within a certain range. To achieve an incremental level in the changes, the various classifiers were trained again more than once, testing different noise ranges, specifically, the ranges are between 0 and: 1, 2, 3, 4, 5 and 6; random numbers chosen within these ranges may seem small, but all the variables were standardised in the preprocessing, therefore they had all originally a mean of 0 and a standard deviation of 1, and after this alteration many instances in the modified variables will therefore become outliers.

5. Results

The acquired evaluation measures are analysed and compared in this section, after going through all the code stages and with a few hours of execution to train the models with the distinct altered datasets.

First, the focus is the construction of the IDS, specifically the performance of the models trained on the legitimate data, with the aim to answer to the first research question. The first metric to consider is accuracy, which is provided separately for the binary and multiple classification tasks in *Figure 6*. The first observable result is that all models achieve good accuracy in the first task, in all the four cases higher than 90%. Looking at the values in the first plot, a little improvement in accuracy is shown as the complexity of the models grows, so first there is the logistic regression with approximately 0.9, then the two ensemble models with about 0.95 and 0.97, and finally the neural network with an accuracy score just below 0.98. The outcome acquired through neural learning is better, but it is not too dissimilar to that of gradient boosting, and so, in terms of computing time and complexity, the latter can also be a good alternative.

Figure 6: Scatterplots of accuracies of the models in the IDS, divided for tasks



As far as the classification into several attack categories is concerned, given the more complex task, worse results are expected, but some models still reach acceptable accuracy here; by looking at the bottommost plot of *Figure 6*, it can be seen that the general trend is similar as in the previous graph, with a difference in the ensemble models, with the random forest performing slightly better than the boosting, but in general the scores grow together with the complexity of the algorithms, in this case even with a much wider discrepancy between the accuracies of the models: the decision tree does not reach an acceptable score, with less than 0.5, the ensemble models far exceeds it with scores of about 0.8, and at the top there is the approximate value of 0.85 for the neural network.

As explained in the evaluation measures section, accuracy alone is not enough to analyse the performance of a classificatory, especially in a problem of this kind. To assess the complete picture of how these models performs in identifying attacks, the confusion matrices and the ROC curves of all of them are visualized in the Appendices B and C, and all their classification metrics are reported in the *Table 2* below. Here, all the values are truncated to the second decimal digit, and in the first column, the models used in the binary task are indicated with (b), while those used in the multiclass with (m).

Table 2: Evaluation measures of the models in the IDS

	Accuracy (test set)	Accuracy (train set)	Precision	Recall	F1 score	AUC
Logistic Regression	0.90	0.90	0.91	0.88	0.89	0.96
Random Forest (b)	0.94	0.95	0.94	0.94	0.94	0.99
Xgboost (b)	0.96	0.99	0.97	0.96	0.96	0.99
Neural network (b)	0.97	0.98	0.98	0.98	0.98	0.99
Decision Tree	0.46	0.81	0.28	0.23	0.20	0.75
Random Forest (m)	0.81	0.82	0.73	0.42	0.41	0.95
Xgboost (m)	0.79	0.89	0.63	0.46	0.48	0.95
Neural network (m)	0.85	0.87	0.72	0.55	0.59	0.97

In the table, the accuracy of the model is reported in two columns, indicating its score both for the train and test set. Regarding precision, recall and F1 score, a value for each of them is assigned to each predicted class, and the value reported in the table refers to the macro average of them, meaning that is does not take in account the number of observations in each class, but only the final value computed for each of them on the test set. Also, in the ROC curves the AUC is computed separately for each class, and again the resulting value reported in the table is the macro average of them, while the specific numbers for each class are reported in the plots within the code.

Considering the complete set of evaluation measures, and looking also at the ROC curves, it is possible to conclude that in both tasks the ensemble models perform way better than the logistic regression or the decision tree, and among the two ensemble models, if considering the accuracy score on train, the gradient boosting outperforms the random forest; however, considering that the neural networks used here have a very simple structure and still get good results, deep learning methods are probably the most suitable for these tasks. When the accuracy values in the train and test sets are compared, it is remarkable to note that during the training phase the gradient boosting model outperforms the deep learning model in both tasks, but not in the test set, indicating that the neural networks are less prone to overfitting. Regarding the other classification metrics, their scores are generally lower in the multiclass task, but the discrepancy from the binary task for them is much bigger compared to the accuracies. Looking at the confusion matrices of the models in the second task, it is possible to understand that such low values in precision, recall, and F1 are due to the fact that the models fail in classifying many of the attack classes, especially those at the end of the matrix, and they assign to the majority of the instances the labels ‘Normal’ or ‘Generic’. Those misclassifications, that do not appear to follow a specific trend since each model has different classes with poor performance, are most likely due to a class imbalance, which means that there are not enough observations in the dataset for each class of attacks to allow the algorithms to classify them all; however, if taken all together as a generic cyber-attack, as in the binary task, the performances are good. The class imbalance also explains why accuracy scores stay high even when other metrics are not, since the classes with higher number of observations are correctly identified.

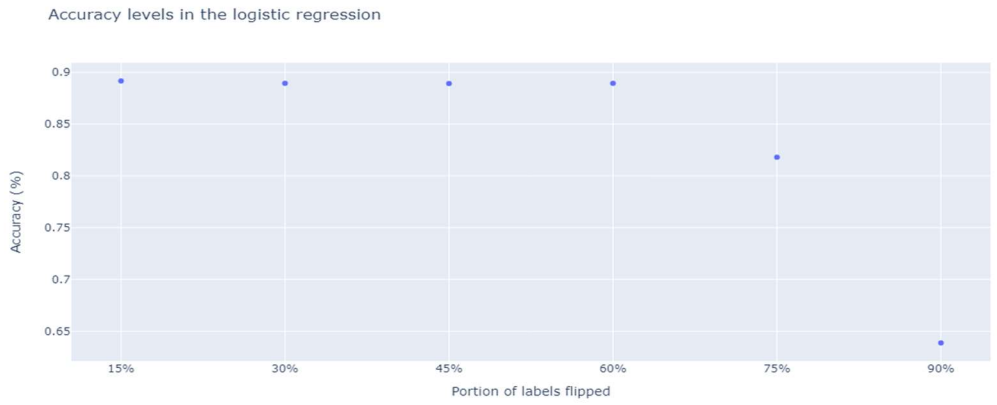
One final detail to observe in those measures is that, in general, all the models have greater precision and lower recall, implying that the rate of false negatives is substantially higher than the rate of false positives. Looking at the confusion matrices again, these numbers may be better understood, since the models properly classify all non-attack occurrences but incorrectly classify many others; this behaviour can be again explained by the class imbalance in the dataset.

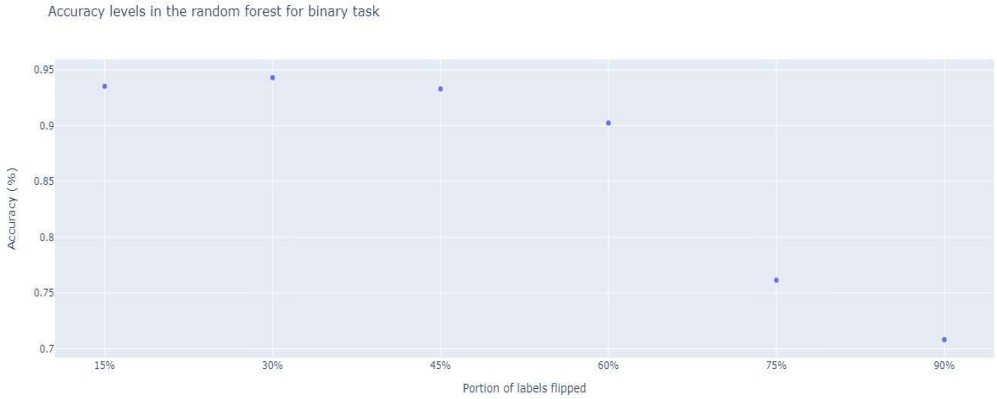
After evaluating the results of the models in developing an IDS, the same measurements produced by the algorithms when trained with the changed datasets will be analysed, in order to answer the second research question.

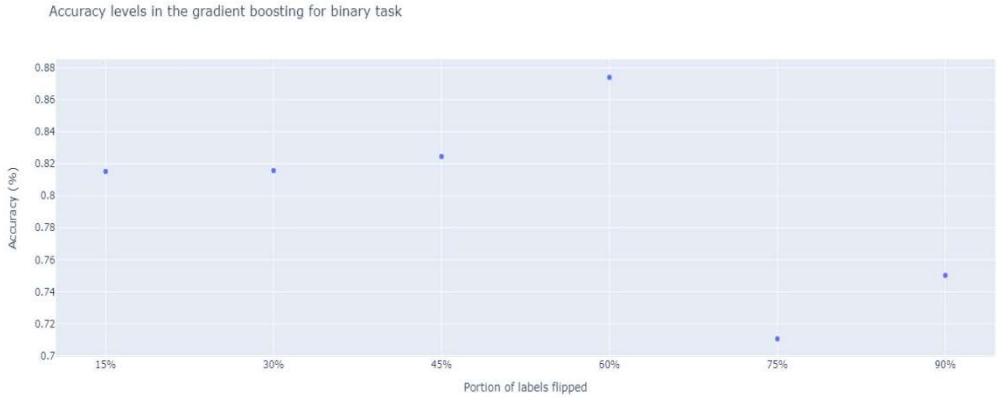
To begin, label flipping with various percentages of the label column shuffled will be looked at as a poisoning method. *Figure 7* shows scatter plots of the models' accuracies on the test set in binary classification, where each point represents the score at a specific flipping rate; is immediately noticeable that, except for few points in the ensemble models, the degradation of performance decreases steadily, meaning that as expected, by increasing the part of mixed labels, the models become less and less capable of producing accurate results. Furthermore, the overall trend in the graphs shows how the algorithms are less impacted by the attack in the initial levels, indicating a general resilience of machine learning to a small portion of wrong labels.

The logistic regression, which set up the lowest accuracy among the models used in this task, appears to suffer less impact between the different levels, since its accuracy remains nearly identical between the flipping rates from 15% to 60%; in any case, this model trained with the most severe level of poisoning shows a decrease in accuracy of 26% when compared to the original one. The impact of the attack is proportionately greater in the more complex models, for instance, the score of 0.63, the lowest ever for binary classification, is obtained with 90% flipping by the neural network, and when compared to the deep learning accuracy value in this task with clean data, its overall decrease is 35%. The impact of the attack with more than half of the label switched is also very strong for the two ensemble models: under this attack, the gradient boosting performs worse than the random forest at low rates of flipped labels, but in the final level the boosting performs better, and these models reach unacceptable accuracies of 0.75 and 0.70, corresponding to a total decrease rate of 24% and 25%, respectively.

Figure 7: Scatterplots of accuracies of binary models with flipped labels







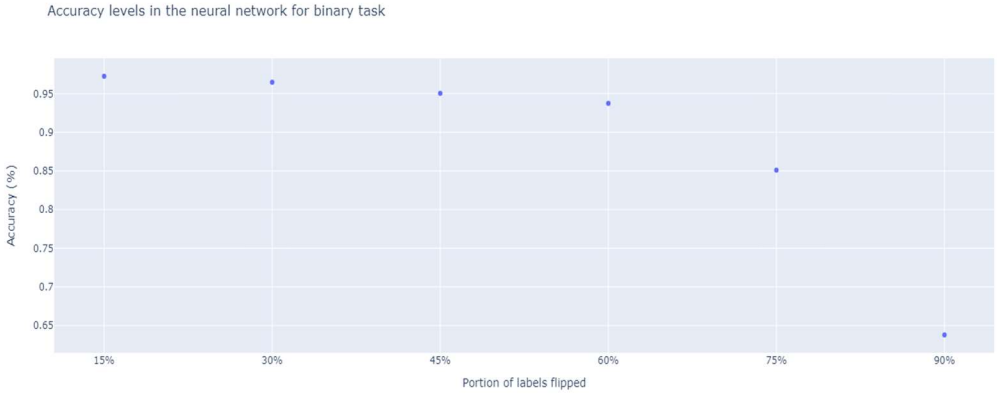


Figure 8: Scatterplots of accuracies of multiclass models with flipped labels

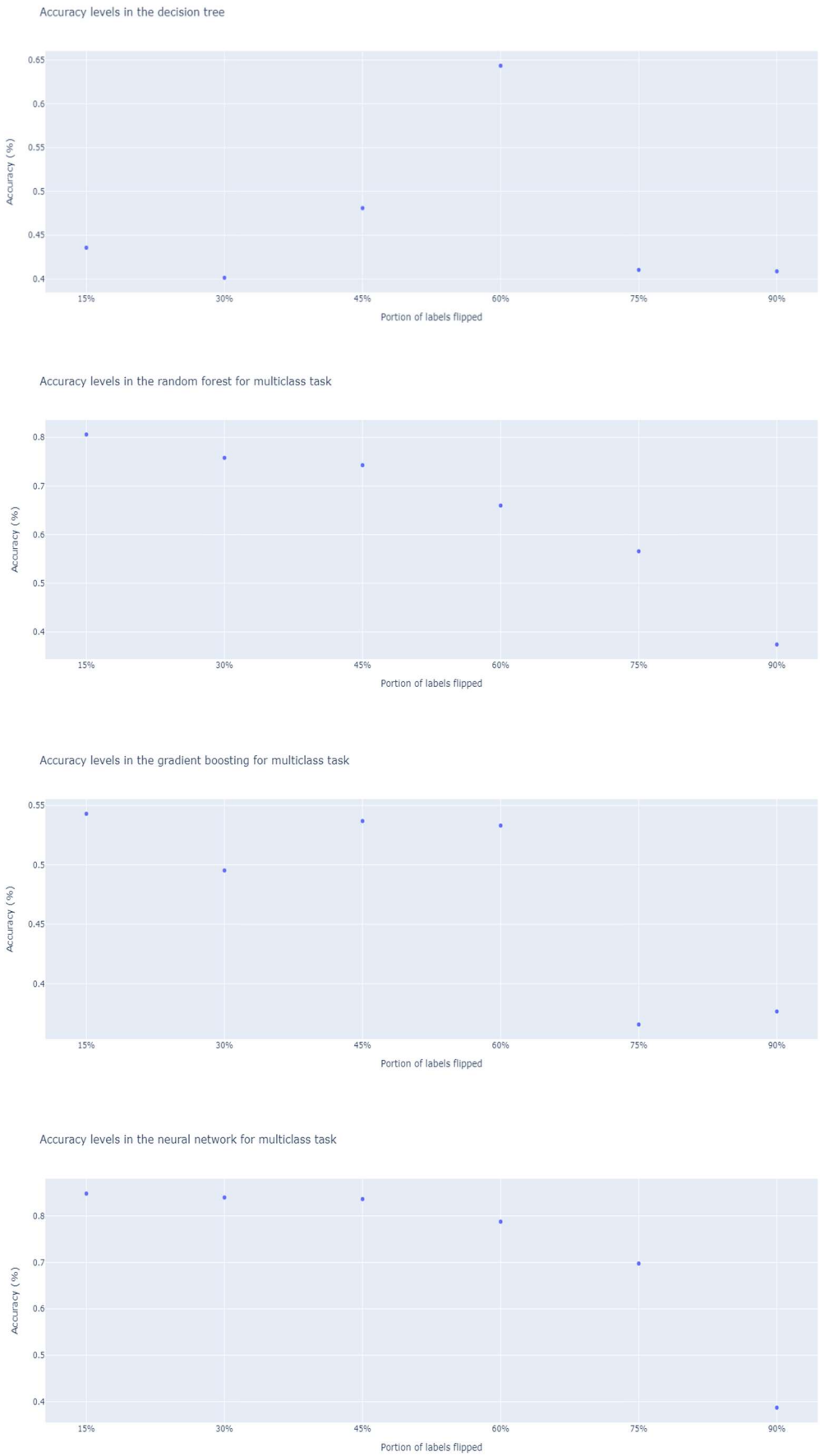


Figure 8 show the same type of scatter plots for the models used in the multiclass task, where the trend of constant decrease in accuracy repeats itself, except for the decision tree and for a couple of points in the gradient boosting, where the accuracies scores in the test does not follow a strong pattern. The decision tree, on the other hand, behaves completely differently than the other models: its highest accuracy is obtained when 60% of the labels are shuffled, and it is interesting to note how here the model achieves a higher score than it did with the clean dataset; label flipping thus has an unusual effect on this model, as can be observed as well from the other points in its scatterplot, and this behaviour is most likely due to the fact that the model, given its simplicity, is in no case suitable for the task. In the more sophisticated models, as in the binary task, the impacts of this attack are disastrous starting from the 60% flipping rate. The random forest and gradient boosting, which had originally accuracy levels of 0.81 and 0.79 respectively, do not manage to exceed 0.4 with the greatest flipping rate. Even in the multiclass task, the neural network suffers the most from the attack, with an almost 50% loss in accuracy when comparing the accuracy gained from the clean dataset to that obtained from the 90% modified dataset.

Table 3 displays the results of all the other evaluation measures for each model in both tasks and for each flipping rate level, employing the identical columns used for the models without any changes to the dataset, again indicated with (b) or (m) if the algorithm was employed in both the classification tasks. To begin, it is crucial to note how the impact of the attack and its rise with the various flipping rates can be better seen by examining the train set's accuracies: Initially, with a low flipping rate, as expected, the accuracies in the train are generally higher than those of the test, but starting from the levels with 30% and 45% of modified labels are examined, the decrease in performance is much more visible in the train; this phenomenon indicates the models' ability to avoid overfitting and to identify the types of classes quite well even if some of those on which they are trained are incorrect. Furthermore, the score for each model in the second column of the table demonstrates a consistent drop, even in the exceptions indicated before.

Table 3: Evaluation metrics of the models with flipped labels

	Flipping rate	Accuracy (test set)	Accuracy (train set)	Precision	Recall	F1 score	Auc
Logistic Regression	0.15	0.89	0.84	0.91	0.86	0.88	0.96
	0.3	0.89	0.79	0.92	0.85	0.88	0.96
	0.45	0.89	0.74	0.92	0.85	0.87	0.96
	0.6	0.89	0.69	0.92	0.85	0.87	0.95
	0.75	0.82	0.64	0.89	0.76	0.78	0.95
	0.9	0.64	0.63	0.77	0.50	0.39	0.95
Random Forest (b)	0.15	0.93	0.89	0.94	0.92	0.93	0.98
	0.3	0.94	0.83	0.95	0.93	0.94	0.98
	0.45	0.93	0.76	0.94	0.92	0.93	0.98
	0.6	0.90	0.70	0.93	0.87	0.89	0.98
	0.75	0.76	0.66	0.86	0.67	0.67	0.97
	0.9	0.70	0.64	0.84	0.60	0.57	0.92
Xgboost (b)	0.15	0.81	0.92	0.86	0.75	0.77	0.91
	0.3	0.81	0.85	0.86	0.75	0.77	0.90
	0.45	0.82	0.78	0.86	0.77	0.79	0.92
	0.6	0.87	0.73	0.91	0.83	0.85	0.96
	0.75	0.71	0.68	0.81	0.60	0.58	0.78
	0.9	0.75	0.66	0.73	0.72	0.72	0.78
Neural network (b)	0.15	0.97	0.91	0.97	0.97	0.97	0.99
	0.3	0.96	0.85	0.96	0.96	0.96	0.98
	0.45	0.95	0.78	0.95	0.94	0.95	0.97
	0.6	0.93	0.72	0.94	0.92	0.93	0.97
	0.75	0.85	0.67	0.87	0.81	0.82	0.93
	0.9	0.63	0.65	0.56	0.51	0.42	0.86
Decision Tree	0.15	0.43	0.73	0.34	0.25	0.23	0.75
	0.3	0.40	0.64	0.32	0.22	0.21	0.74
	0.45	0.48	0.56	0.40	0.22	0.20	0.78
	0.6	0.64	0.48	0.40	0.30	0.30	0.86
	0.75	0.41	0.41	0.36	0.20	0.20	0.69
	0.9	0.40	0.36	0.15	0.13	0.11	0.76
Random Forest (m)	0.15	0.80	0.74	0.47	0.39	0.39	0.94
	0.3	0.75	0.65	0.54	0.37	0.37	0.94
	0.45	0.74	0.56	0.48	0.32	0.33	0.93
	0.6	0.65	0.48	0.51	0.26	0.26	0.94
	0.75	0.56	0.40	0.41	0.20	0.19	0.92
	0.9	0.37	0.36	0.38	0.11	0.07	0.82
Xgboost (m)	0.15	0.54	0.79	0.48	0.36	0.37	0.88
	0.3	0.49	0.69	0.59	0.32	0.32	0.88
	0.45	0.53	0.59	0.56	0.31	0.32	0.91
	0.6	0.53	0.50	0.54	0.29	0.29	0.83
	0.75	0.36	0.43	0.41	0.16	0.17	0.82
	0.9	0.37	0.39	0.19	0.12	0.10	0.77
Neural network (m)	0.15	0.84	0.77	0.69	0.52	0.55	0.95
	0.3	0.83	0.67	0.71	0.51	0.54	0.94
	0.45	0.82	0.58	0.67	0.48	0.51	0.93
	0.6	0.79	0.49	0.62	0.39	0.41	0.90
	0.75	0.69	0.41	0.54	0.31	0.33	0.88
	0.9	0.38	0.37	0.25	0.12	0.09	0.77

Among the various evaluation measures, the AUC has the least impact: in general, this measure also follows the trend of constant decrease as the flipping rate increases, but the values from one level to another change much less than those of accuracy; it should also be noted that the greatest variation of AUC occurs in the more complex models, specifically gradient boosting and neural networks, and in general it is possible to observe a significant decrease in AUC almost only in the highest flipping rates, particularly the last one. The ROC curves and confusion matrices of the various models subjected to label flipping at the 90% level can be found in the Appendices F-I, where it is possible to witness how, for those used in the multiclass task, the last types of attacks and so the less common classes, have generally lower scores. Regarding precision and recall, except for the same points in which some models had unusual decrease in accuracies, the decrease of the values proceeds with the increase of the flipping rate in the various models; it is also interesting to notice that to those unexpected points with higher test accuracy outside the decreasing trend, is associated an higher precision, while the recall continuously decrease with the higher flipping rate levels. As in the basic IDS algorithms, the precision has generally higher values than the recall; additionally, when observing the various levels of flipping, the precision' scores decrease proportionally faster than those of the recall. The F1 score, which is a weighted average of the two preceding measures, similarly reveals an overall decline in performance, for instance, in the multiclass neural network falls from 0.55 to the insignificant value of 0.09.

In the following paragraphs the outcomes of the second method used to simulate data poisoning will be investigated, to understand the effects of an addition of random noise within a specific range, on all occurrences but only on the values of ten of the predictors.

To begin, the scatter plots showing the accuracy of the models in the numerous attempts to modify the dataset are reported, in a manner similar to the other technique, with the noise range on the x-axis, indicating the greatest number of random noises that might be introduced to any instance. The scores of the models utilized in the binary task are presented in *Figure 9*, and the first thing to note is that there is no pattern of decreasing accuracy with the rise of noise range. In general, the initial levels generally achieve higher accuracies in all the models, but looking at the y-axis of all the graphs, the differences between the various dataset adjustments are extremely tiny. Analysing the graphs one by one, it is shown that the logistic regression accuracies all remain at 0.89, the random forest accuracies have slightly changes remaining between 0.93 and 0.95, the gradient boosting oscillates between 0.96 and 0.97, and the neural network also remains fixed at 0.97, but with a very slow constant decrease along with the levels, appearing to be the model that follows more the expected trend. *Figure 10* shows parallel graphs for the models used in multiclass classification, and again, no common pattern

can be identified, and the differences in accuracy between the various noise range levels are small. The decision tree appears to be the model that has the biggest changes in accuracies' values, with a range between 0.81 and 0.75, and it is surprising to note that these values are much higher than those of the model trained with the clean dataset; the random forest also, compared to the other algorithms, shows great variations between one level and another, even if its scores remain between 0.79 and 0.74, while the boosting accuracies remain between 0.81 and 0.84, and the neural network have almost irrelevant variations, remaining always 0.84 and below 0.85. Furthermore, under this sort of assault, the xgboosts outperforms the random forests, while the deep learning models remain those with higher accuracies.

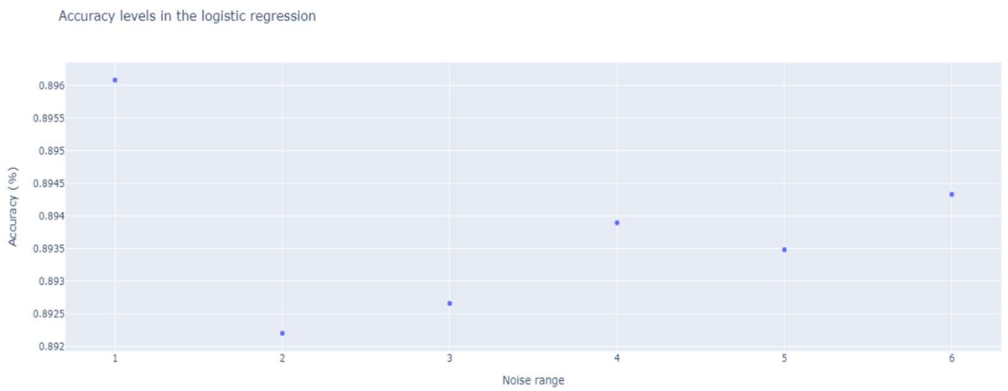
All the classification metrics of the various models for the two tasks are presented in *Table 4*, as was done for the basic IDS models and the previous poisoning strategy, with the noise range levels indicated in the first column. Looking first at the accuracy in the train sets, it can be noted that there are essentially no changes in values across the different levels in all models of both tasks, with the exceptions of the random forest in the binary task, which drops after the first level from 0.95 to 0.94, and of the neural network in the multiclass task, which oscillates between 0.86 and 0.85. This result is not surprising after examining the accuracies on the test set, but the scores on train show even less variation, even for the models that had some accuracies' decrease on the test, which as previously mentioned were some of the simpler ones, for instance the two random forests and the decision tree; this may indicate a tendency of these models to be less resistant to this type of attack, as the algorithm's training has adapted well to the changes in value, however leading to slightly worse results when evaluated on the test set, just as expected by this method of poisoning. It is also interesting to note that in both the classification tasks the boosting is the model obtaining higher accuracy scores in the training, even if when this measure is computed on the test set, it is often outperformed by the neural network.

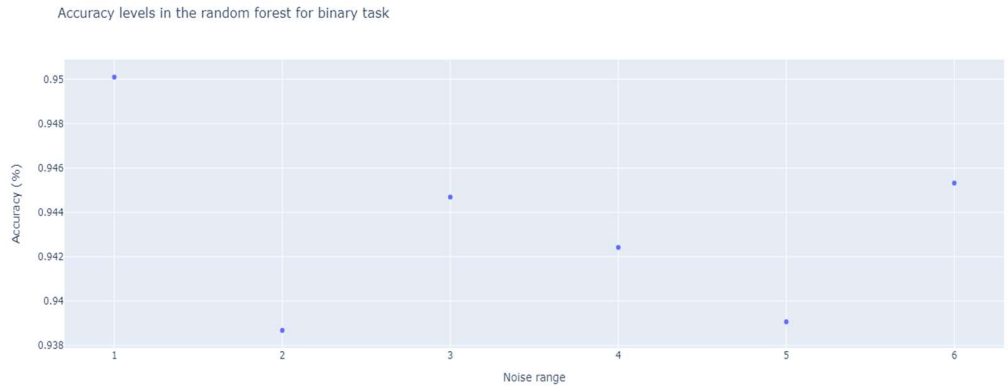
Observing Precision, Recall, and F1 score computed on the test set, on the other hand, it is possible to notice some slightly greater variations in values in some models, as can also be seen in the confusion matrices of those with the highest range of noise in the Appendices J and K, which present slightly different numbers of false positives and negatives from time to time; however, none of the models shows a constant decreasing pattern, meaning that the two measures representing the two types of statistical errors never fall with the noise's levels. The precision in the logistic regression remains between 0.91 to 0.90, and the recall goes from 0.87 to 0.86, but the F1 remains constant at 0.88. The random forest in the binary task has precision values between 0.95 and 0.94, and recall values between 0.94 and 0.93, which leads to an F1 with similar tiny deviations; the same algorithm

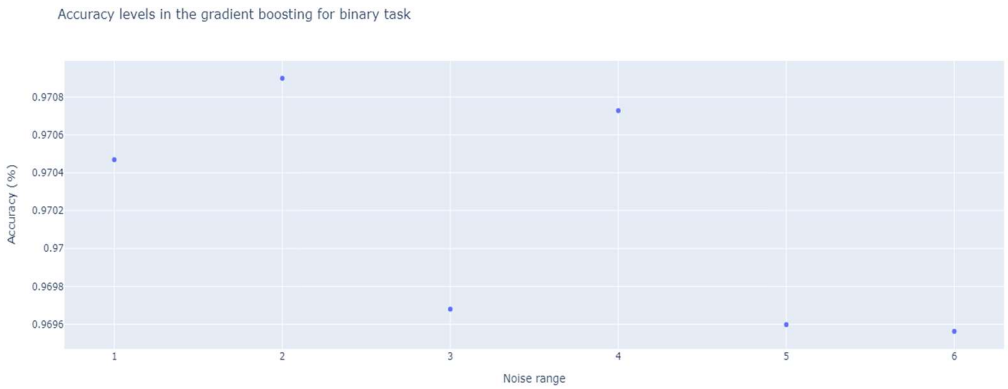
in its multiclass form has precision values between 0.60 and 0.43, which are way higher than the recall values, more similar to those of the F1, between 0.38 and 0.34. The gradient boosting in the binary task maintains precision and F1 scores of 0.97, while the recall drop to 0.96 at some noise levels, and in the multiclass classification the model values drop to the range 0.71-0.66 for precision, 0.53-0.51 for recall, and 0.56-0.53 for F1; when discussing the performance of the neural network, the binary task values in these three measures remain fixed at 0.97, while in the multiclass task there is slightly more variations as in the other metrics, with a precision between 0.66 and 0.61, in which is possible to note lower values respect to the gradient boosting, while recall and F1 remains slightly above 0.50.

Regarding the ROC curves, which can be consulted in the Appendices L and M, and the AUC of the various models, the numbers per model change very little between the different levels of the noise range, and if present, the variations roughly follow the patterns described by the other evaluation measures described.

Figure 9: Scatterplots of accuracies of binary models with random noise







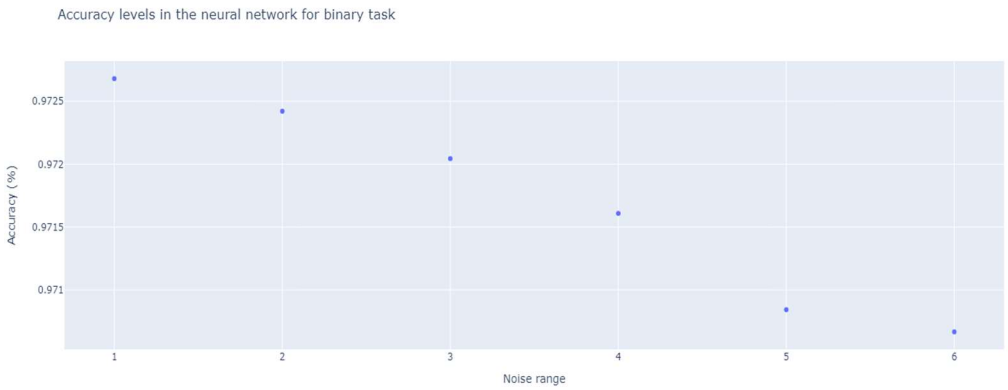


Figure 10: Scatterplots of accuracies of multiclass models with random noise

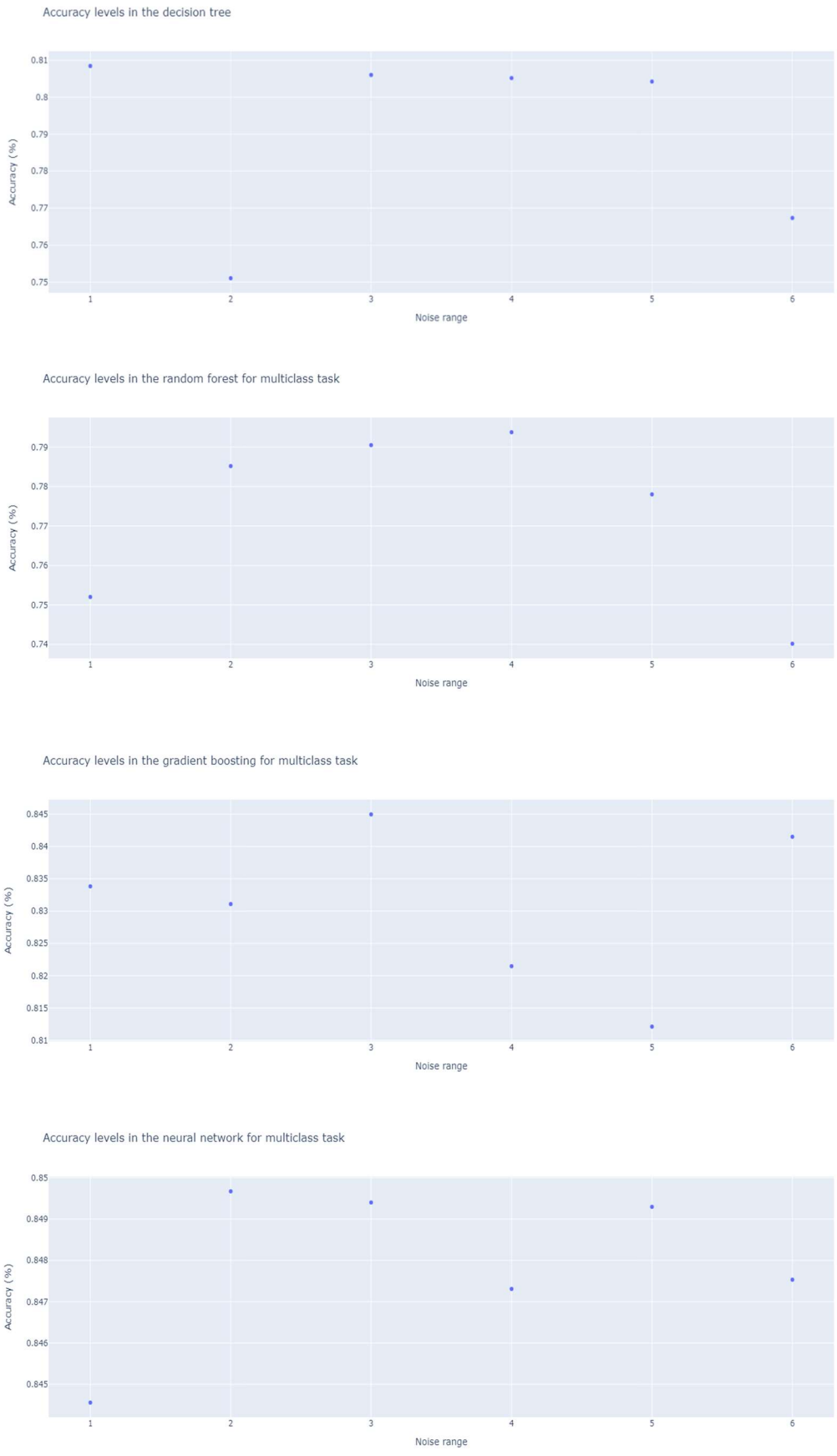


Table 4: Evaluation metrics of the models with random noise

	Noise range	Accuracy (test set)	Accuracy (train set)	Precision	Recall	F1 score	Auc
Logistic Regression	1	0.89	0.89	0.91	0.87	0.88	0.95
	2	0.89	0.89	0.90	0.86	0.88	0.95
	3	0.89	0.89	0.90	0.86	0.88	0.95
	4	0.89	0.89	0.90	0.87	0.88	0.95
	5	0.89	0.89	0.90	0.87	0.88	0.95
	6	0.89	0.89	0.91	0.87	0.88	0.95
Random Forest (b)	1	0.95	0.95	0.95	0.94	0.95	0.99
	2	0.93	0.94	0.94	0.93	0.93	0.98
	3	0.94	0.94	0.94	0.94	0.94	0.98
	4	0.94	0.94	0.94	0.94	0.94	0.98
	5	0.93	0.94	0.94	0.93	0.93	0.98
	6	0.94	0.94	0.94	0.94	0.94	0.98
Xgboost (b)	1	0.97	0.99	0.97	0.96	0.97	0.99
	2	0.97	0.99	0.97	0.97	0.97	0.99
	3	0.96	0.99	0.97	0.96	0.97	0.99
	4	0.97	0.99	0.97	0.97	0.97	0.99
	5	0.96	0.99	0.97	0.96	0.97	0.99
	6	0.96	0.99	0.97	0.96	0.97	0.99
Neural network (b)	1	0.97	0.98	0.97	0.97	0.97	0.99
	2	0.97	0.98	0.97	0.97	0.97	0.99
	3	0.97	0.98	0.97	0.97	0.97	0.99
	4	0.97	0.98	0.97	0.97	0.97	0.99
	5	0.97	0.98	0.97	0.97	0.97	0.99
	6	0.97	0.98	0.97	0.97	0.97	0.99
Decision Tree	1	0.80	0.83	0.63	0.43	0.45	0.93
	2	0.75	0.83	0.49	0.38	0.39	0.92
	3	0.80	0.83	0.53	0.42	0.43	0.94
	4	0.80	0.83	0.53	0.42	0.43	0.94
	5	0.80	0.83	0.52	0.42	0.43	0.94
	6	0.76	0.83	0.51	0.40	0.42	0.92
Random Forest (m)	1	0.75	0.81	0.48	0.34	0.34	0.94
	2	0.79	0.81	0.56	0.36	0.37	0.95
	3	0.79	0.81	0.60	0.38	0.38	0.97
	4	0.79	0.81	0.46	0.37	0.37	0.95
	5	0.77	0.81	0.46	0.35	0.35	0.95
	6	0.74	0.81	0.43	0.34	0.36	0.94
Xgboost (m)	1	0.83	0.90	0.69	0.53	0.56	0.97
	2	0.83	0.90	0.71	0.53	0.56	0.97
	3	0.84	0.90	0.70	0.52	0.55	0.97
	4	0.82	0.90	0.67	0.51	0.53	0.97
	5	0.81	0.90	0.67	0.51	0.53	0.96
	6	0.84	0.90	0.66	0.52	0.55	0.97
Neural network (m)	1	0.84	0.86	0.61	0.53	0.54	0.96
	2	0.84	0.86	0.63	0.52	0.54	0.96
	3	0.84	0.86	0.66	0.53	0.56	0.96
	4	0.84	0.85	0.68	0.51	0.52	0.96
	5	0.84	0.86	0.66	0.52	0.55	0.97
	6	0.84	0.85	0.65	0.51	0.53	0.96

6. Preventive Measures

After conducting the practical analysis of this work and having the results of the impacts suffered by an IDS built through machine learning when subjected to two simple data poisoning simulations, in this chapter it is time to explore the possible preventive measures that can, and often must be developed to counter the threat of possible data tampering. The attack simulations explained in the previous chapters can give a better idea of the objectives and techniques used by a potential attacker, and in real life situations, a malicious user with computer skills and the ability to access the data of a system can lead to really serious consequences, in even more subtle and complex ways than those explored in the technical part of this paper; as a result, it is essential to look at the situation from the point of view of those who are entitled to protect systems from these threats, and to delve into the several possible defence strategies, mentioned in the studies considered in the literature review. Some of the most popular and effective preventive measures will now be explained, giving them context, and imagining how they would handle the data modification presented above, analysing their advantages and limitations.

When constructing a machine learning system, there are numerous components and stages to consider in order to maintain its security. The first of these, and perhaps the most crucial when addressing the issue of poisoning, is to ensure the quality of the data. There are several aspects to consider when it comes to data quality, since the entire preprocessing stage through which they go through before being fed to the algorithms is designed to improve quality, but from a cybersecurity standpoint, what is most meaningful is data authentication, which is the process that ensures that the data have not been altered or compromised during their transmission, storage, or processing. It is also possible to identify two major components in the data authentication process: data integrity, which refers to the assurance that the data remain unchanged throughout their lifecycle, and data authenticity, which involves verifying the origin of the data and ensuring that they come from trusted and legitimate sources.

Verification of data sources must be the foundation of any secure system and given the massive amount of data that businesses today have to deal with, and the fact that those data are sent to them from a variety of sources, one of the best ways to ensure data integrity is to use cryptographic techniques, such as hash functions or so-called checksums. Hash functions generate fixed-length digests based on the data's content that are unique to the data and will result in a different hash if the data is changed, these functions are therefore feasible to determine whether the data have been modified by comparing the computed hash of the received data to the original hash provided by the sender. Checksums, on the other hand, are values calculated from the sum of all data bytes in a file,

and, like hash functions, changes in the data will result in different checksums. In terms of data authenticity, here again cryptography provides very valid methods to ensure the origin of the data, one of which is the use of keys associated with the identity of a specific entity, to be used for example in a system of digital signature, which uses asymmetric cryptography to verify the data and works briefly as follows: the sender uses their private key to encrypt a hash of the data, then the receiver can use the sender's public key to decrypt the signature and compare it to a recalculated hash of the received data.

The methods outlined above are applicable in many systems for a wide range of uses, for instance, it is sufficient to consider that hash functions and digital signatures constitute the foundation of all blockchain technologies. Furthermore, given their applicability to any type of data, the usage of such techniques in IoT systems is not difficult to imagine: one possible example is a connected healthcare monitoring system, in which it is critical to ensure the security and authenticity of health-related data transmitted between wearable health devices and a central healthcare server, and this goal is achieved through an integrity verification with hash function used in conjunction with a public key infrastructure; moreover, to ensure that the people who have access to such data are only those who are allowed to do so, additional authentication mechanisms, such as the use of credentials and a secondary authentication factor, can be introduced.

The main benefit of implementing these data authentication measures is that they provide a strong security foundation, without which any system would be extremely vulnerable, and this leads to increased data trustworthiness; additionally, a great reason to implement these defences is early detection, which means that by actively monitoring data integrity and authenticity, suspicious activities or data tampering attempts can be detected in the early stages of the whole data science process. There are currently several limitations to consider regarding these measures, such as the reliance on data sources, which if themselves compromised, could undermine the authenticity of the data, along with the fact that implementing cryptographic measures is very resource intensive and not affordable for many organizations. It should also be noted that data authentication methods may be ineffective against novel or sophisticated data poisoning attacks that leverage weaknesses not addressed by existing authentication approaches.

Outlier detection is an operation that can be performed on data before using it in machine learning, which is typically inserted between the various steps of the preprocessing stage and can be highly beneficial in detecting data poisoning. Outliers are points in a dataset that deviate significantly from the rest of the data, and regardless of the task, it is critical to identify the presence of these points and treat them appropriately, as they frequently represent anomalies, can induce bias, or are simply better

eliminated for the sake of model accuracy. Outlier detection, which aims to find anomalous points, is extremely useful in identifying malicious data injection by a potential attacker: as seen in the practical part of this work, particularly in the second method used to simulate an attack, poisoning attacks frequently involve introducing noisy or incorrect data points to degrade the performance of a model, and thus outlier detection can help identify and remove such points, ensuring that the model's performance is not degraded. Outlier detection can also be utilized dynamically during model deployment to continuously monitor incoming data: if the system identifies an unexpected surge in the number of abnormal inputs, it can generate alarms and briefly suspend the model's execution for additional investigation. Even for this defence technique, there are numerous scenarios of complex real-world systems in which its application becomes critical, for example, consider a fleet of trucks transporting specific materials that, using IoT sensors, monitors some parameters to be respected, such as temperature or humidity: if one of the vehicles is altered in a malicious way, the outlier detection system would immediately detect the anomaly.

Among the primary benefits of implementing outlier detection are improved model performance and bias mitigation; in terms of its specific use as a preventive measure against poisoning, outlier detection allows for early detection, so to prevent the danger before the model training phase, and at the same time it can be implemented as a real-time monitoring method. In terms of limitations, it is needed to recognize that there is a need for contextual interpretation to identify outliers, given that unusual points can occur naturally in datasets, and that with some types of scaling or high dimensionality of the data, which is frequently required to represent data with complex distributions, the technique is not always effective.

Another intriguing option to investigate is data augmentation, which was not expressly created as a preventive measure for data modifications but is very beneficial in minimizing the effect of potential data poisoning. The purpose of data augmentation is to increase the variety of the training dataset, creating new, slightly modified examples from the original data by applying various transformations, in order to improve model generalization and robustness. Image recognition or categorization are typical tasks in which augmentation techniques are employed, in which pictures are rotated, flipped, resized, or subjected to other alterations to replicate real-world differences among the standard data and add variability. Because of its function, that is to incorporate a higher level of diversity and randomness into the data, data augmentation can become an effective preventive measure against poisoning, not only because it makes the model more accurate and capable of handling higher variability, but also because, given that these attacks frequently try to insert a specific type of malicious pattern on which to train the models, these will appear more diluted in the augmented

dataset, making the impact of the data modification weaker. Although not a true barrier, this approach is quite beneficial for minimizing the symptoms of a poisoning, and one of its key advantages is its ease of implementation. In a similar vein to the preceding strategies, the more targeted and subtle the attack is, the less helpful augmentation will be in fighting it.

Among all the techniques that can be used in machine learning to make systems more robust and secure, one in particular is well suited to countering data poisoning: adversarial training, which, as the name implies, derives from adversarial attacks and acts as a defence against them. The adversarial training process can be seen as a voluntary poisoning of the dataset, obtained through the creation of adversarial examples using optimization techniques: these are often much more specific and complex than the methods used to simulate the attack seen in this paper, such as the Fast Gradient Sign Method (FGSM) or the Projected Gradient Descent (PGD), to name a few. The adversarial instances are added to the training dataset, in form of unnoticeable alterations that cause the model to make inaccurate predictions, and its resilience is improved by retraining it iteratively with fine tuning capable of managing the changes. Adversarial training is a very versatile technique that can be incorporated into data science architectures of any sector, such as an intrusion detection system, which, as previously explained, is widely utilized in IoT domains as well as other infrastructures based on the analysis of internet traffic. As a result, the adversarial training method is among the most effective in boosting systems' robustness, and it is also compatible with any kind of model. Being a method based on data modification, it may be the best to mitigate the danger of data poisoning, but it is also very specific, and would be ineffective, for example, in facing a label flipping; among its limitations there is also a significant increase in the need for computational resources, which is due to the fact that a very meticulous fine tuning must be included in its process.

Another technique worth mentioning, which is frequently used during model training and has the main goal of selecting the best model architecture and tuning hyperparameters to improve generalization, is the use of a validation set: this term refers to a small part of the dataset kept separate from the train, on which to evaluate the model's performance, observe if the results are satisfactory, and if they are not, iteratively adjust hyperparameters. A supplementary split in addition to train and test sets may appear unnecessary on a theoretical level, but in practice, the test may not be representative of the entirety of the data, or it may only be utilized after the system is deployed, and fine-tuning is obviously required first. The validation set can indirectly help mitigate the impact of poisoned data by making the models more robust in general and, more importantly, by identifying if the model's performance drops dramatically on the validation data due to the inclusion of attacked training samples. Because its primary function is not that of a preventive measure, this technique is

not always effective, as it can detect the presence of poisoned data only if the attack is quite severe in terms of performance loss, and furthermore, the validation set itself may be reachable by the attacker and poisoned.

A brief examination of the various prevention measures listed in this section demonstrates how useful and effective data manipulation techniques are in preventing potential cyber attacks, and how, thanks to growing innovation, these methods are becoming increasingly effective and widespread; however, at the same rate of research in these areas, the capacity of potential attackers grows, and the statistical rules that govern data science techniques, no matter how complex, can be circumvented, and this is where the human aspect, or the concept of Human-in-the-Loop, comes into play and remains vital. What can truly make a difference in IT security, particularly in the case of zero-day assaults, is the systems' monitoring by experts at various stages of the process, who, through domain knowledge and expertise, can fully use the potential of the machines and support them in any instance of tampering. The human aspect is not a true preventive strategy, but it is what best allows to reduce damage in the case of an emergency, by establishing an iterative feedback loop in which human experts cooperate closely with the machine learning system.

Data security is a subject that all businesses must face, and as seen, just for a specific form of attack several possible protection measures have been developed, based on different necessities. However, each of them has significant limits, through which an attacker with considerable adaptability might hunt for vulnerabilities in the system to exploit. Despite the advancement of security systems, it is not possible to be completely safe from cyber risks, and data poisoning is one of these unpredictable threats: as highlighted in this section, some preventive measures can be excellent on one type of data modification but can on the other hand be completely ineffective on others. The wisest thing to do, especially in sensitive areas where the precision of machine learning systems is critical, is therefore to incorporate more than one of these safeguards together, to make the models as secure as possible.

In modern infrastructures, it is then critical to first have a clear perspective of the organization's risks, and attempt to create an adequate defence network, while also being prepared to react in the case of a breach and mitigate as much as possible the damages. In order to create such defence systems that are robust and ready to respond to all types of threats, more and more organizations are relying on digital twins, which are created by combining AI methods with organizational architectures in order to improve cybersecurity measures and overall system resilience. By simulating the consequences of various actions, digital twins are extremely effective in developing response strategies, and by incorporating IDSs into their development, it would be possible to achieve a comprehensive defence system adequate for each stage of possible attacks, from prevention to recovery. The ability of digital

twins to simulate system behaviour by providing accurate contextual information can be very useful to an IDS in identifying the type of activity considered normal, and thus to identify deviations that may indicate security threats more accurately; additionally, the IDS can adapt its detection algorithms to the evolving behaviour patterns in the digital twin, and the digital twin can learn from real-world security incidents, allowing for the development of a continuous learning feedback loop in cybersecurity practices.

7. Conclusions

Understanding how data science techniques can improve security measures is critical for effective risk management, in an era where cyber threats are always changing and becoming more sophisticated: as organizations increasingly rely on digital infrastructure and data-driven decision-making, the integrity of their data and the effectiveness of their security systems are critical. Conducting a study in this domain can therefore help in providing executives with the knowledge and tools needed to make informed decisions that strengthen cybersecurity strategies and ensure the resilience of their operations in the digital age. Moreover, understanding the interaction between intrusion detection systems and data poisoning is critical for managers tasked with protecting their networks and sensitive data. In this last chapter, the overall conclusion that can be draw from this paper are summarized.

The first part of the practical investigation conducted in this study demonstrated how an efficient intrusion detection system can be built using machine learning algorithms trained on a suitable dataset. Several conclusions can be drawn from the comparative analysis of the different models, the first of which is that binary classification, that is, the distinction between normal and suspicious Internet traffic activity, is a goal that can be achieved with good results without too much difficulty. All four models examined attained at least 90% accuracy in the binary task and performed well in the assessment measures; among them, the logistic regression is the least effective due to the nonlinearity of the data, while the two ensemble models and neural network are more efficient. Moving on to the multiclass task, meaning the recognition of the type of cyber-attack in each specific case, the complexity increases, and as a result the performance of the models suffers; here, a simple model such as the decision tree fail in completing the task with acceptable performance, however, the random forest and the neural network are not completely ineffective, as they achieve respectively 81% and 85% accuracy, despite failing to recognize some classes with few instances. The differences between the various models are therefore obvious, and for a job of this type, it is necessary to use complex algorithms, sacrificing some interpretability. In the future, we should expect outstanding outcomes in the development of these types of systems, which can be achieved by fine-tuning the ensemble models or through more complicated and longer trained neural networks.

Proceeding with the practical analysis, it was possible in this work to observe the impact of two simple modifications on the dataset, simulating data poisoning, on the performance of the models initially observed. As explained in detail in the results section, by using label flipping and gradually increasing the portion of the modified dataset, a constant decrease in model performance is observed, and the effects of this attack appear destructive on the quality of the models' predictions; on the other hand,

in the second method used, that is the addition of noise in some of the variables of the dataset, there is no significant decrease in performance, and furthermore, as the range in which this noise is produced increases, the decrease in the quality of the models is not incremental. With a little technical context, it is not surprising that the first method produces such noticeable results: the association of the respective labels with the observations of the dataset is what allows any supervised model to learn, and eliminating this association in more than half of the instances makes the training of the algorithms completely inefficient. One noteworthy finding in these simulations is that neural networks are more heavily impacted by this first attack than ensemble models, while still producing the best results in nearly every case.

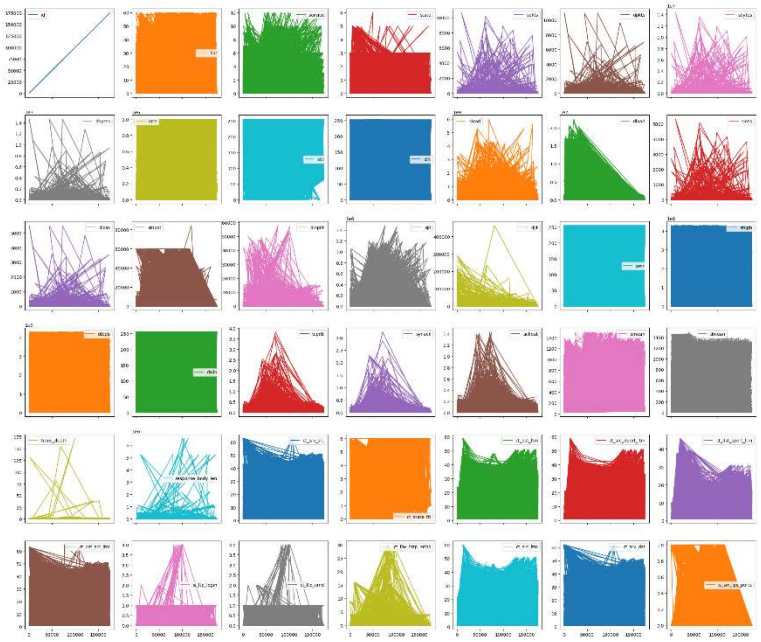
Although the addition of noise is a change that is quite similar to actual data poisoning, it does not produce astonishing results when carried out as done in this study, most likely because it is accomplished by adding random values, and only on 10 variables of a dataset that comprises more than 40. Furthermore, the difference between the various levels in this second method is represented only by an increase in the range in which the possible added random value is taken, and the results show that these increases have no significant impact on the evaluation measures in the various algorithms. Future studies could investigate towards utilizing noise to modify a dataset in a more precise way, such as forcing the model to predict a specific type of erroneous class, or perhaps applying this modification to all the variables.

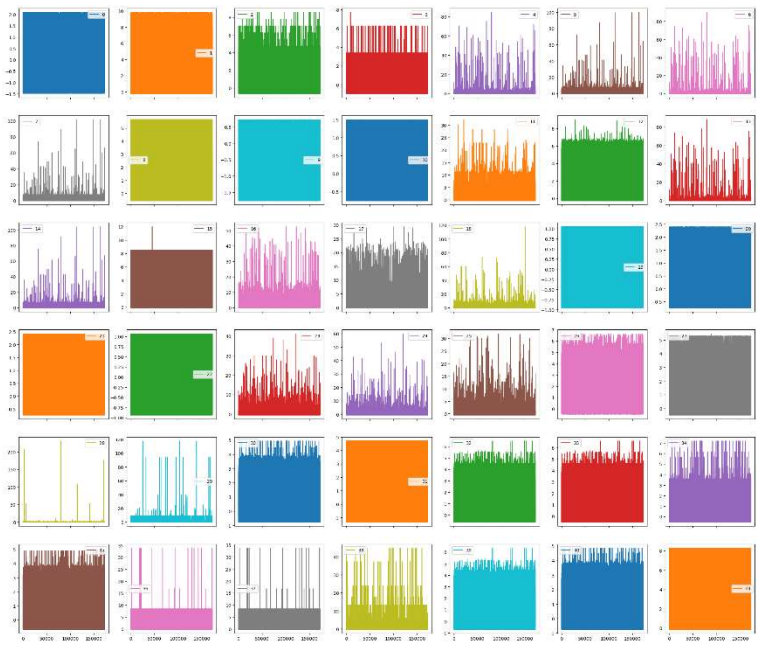
Following the investigation of the tangible consequences of data changes on models, some of the most widespread prevention strategies were presented in the previous chapter to provide a comprehensive picture of a potential attack and defence scenario. Many of the strategies discussed are themselves based on data science, proving once again the enormous flexibility of this field of study, and its effectiveness in many environments, as illustrated by the examples provided, entailing the usage of IoT technologies. In the future, it would be interesting to put these various preventive measures to the test, in a practical way, on the various possible methods of attacking a dataset, resulting in a complete and measurable framework useful for associating the specific risks of each organization with the development of a suitable defence system.

8. Appendices

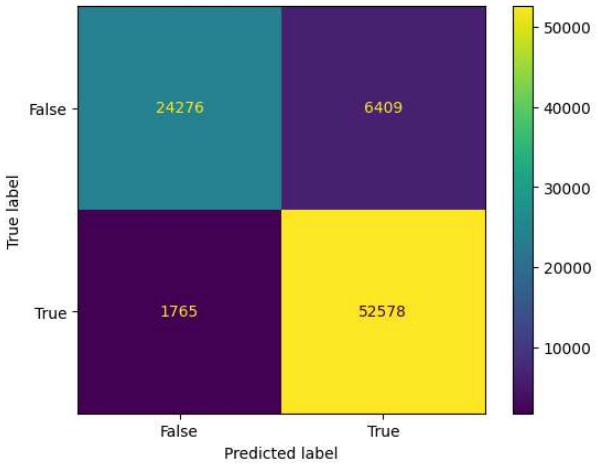
In this section are reported the appendices, a set of plots, that except for the first, show the Confusion matrices and the ROC curves of the original IDS built with the different models, and then those of the highest level of both methods of poisoning, to observe in a graphical way the decrease performance. The plots for each of the other level of poisoning could be find in the code.

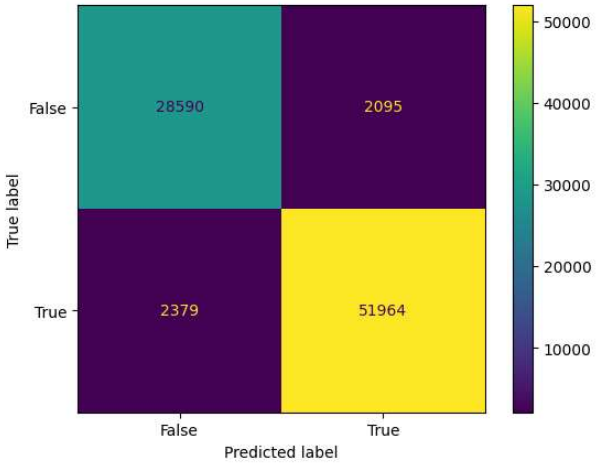
Appendix A: Line plots of the dataset divided for features, before and after standardization.

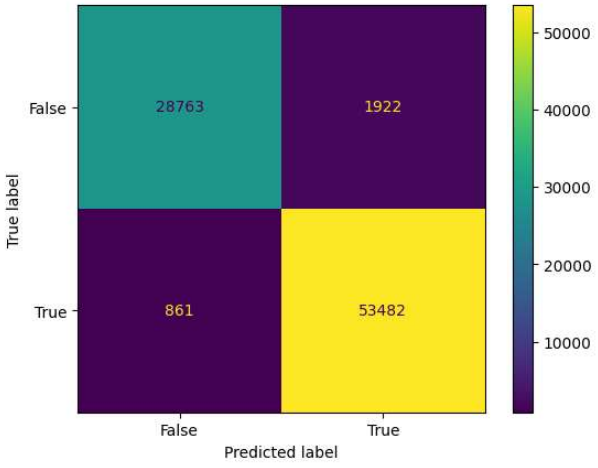


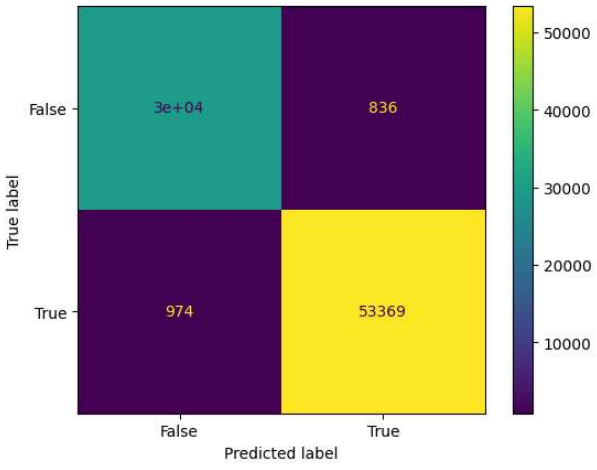


Appendix B: Confusion matrices of binary models (logistic regression top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

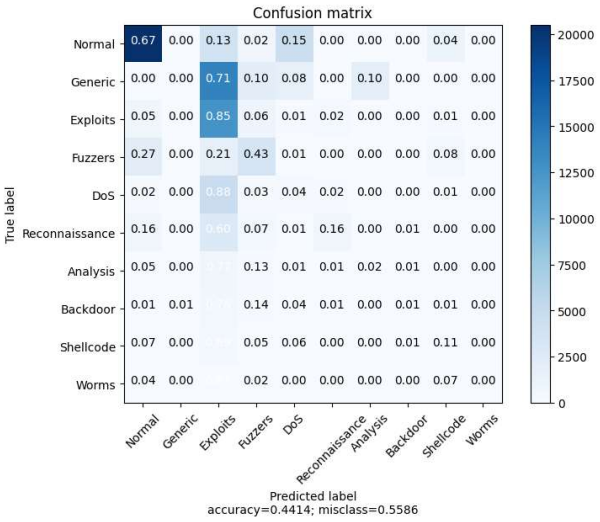


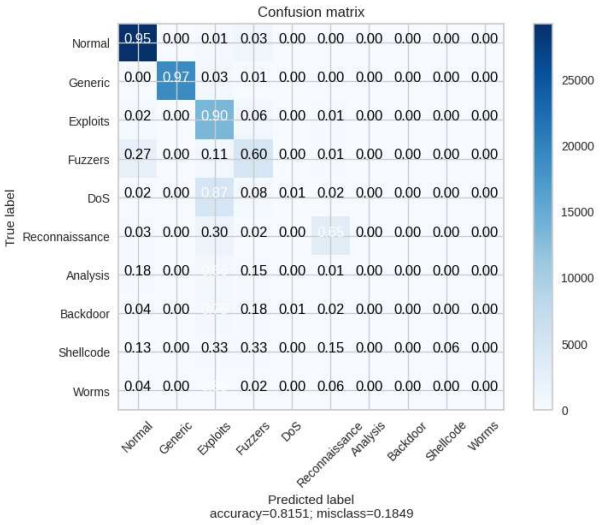


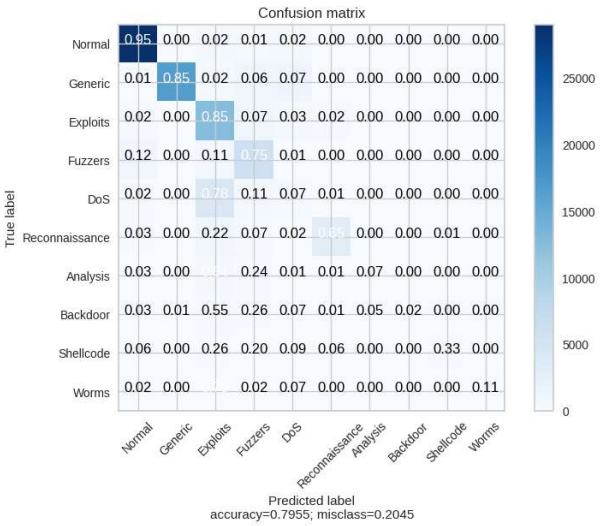


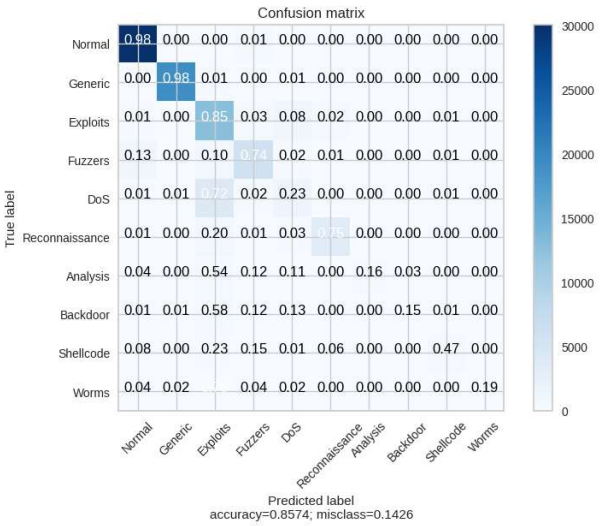


Appendix C: Confusion matrices of multiclass models (decision tree top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

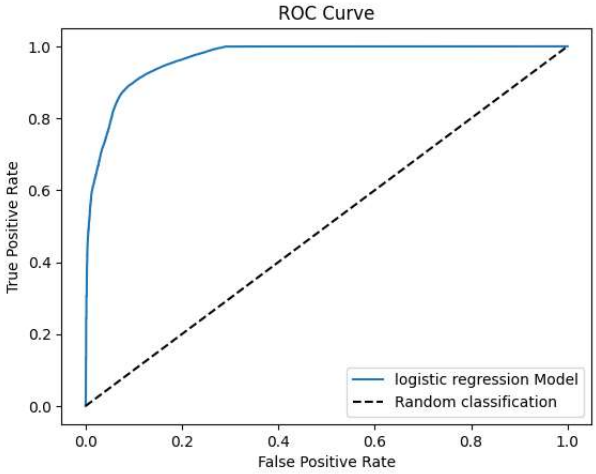


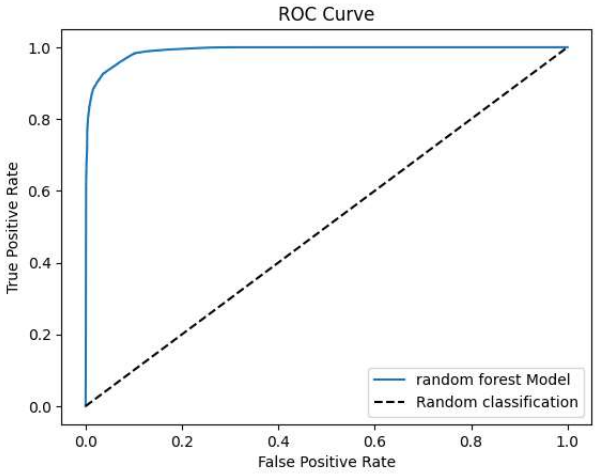


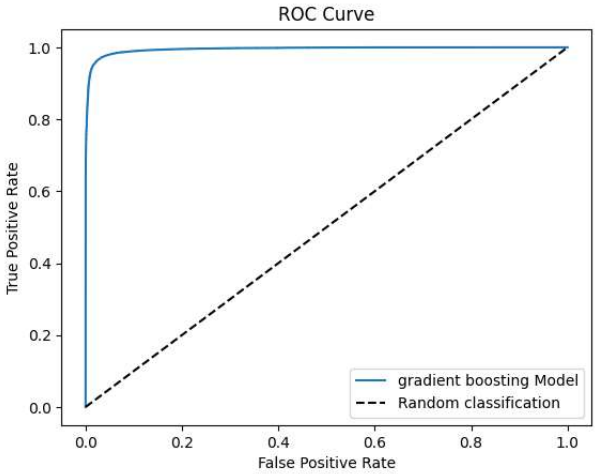


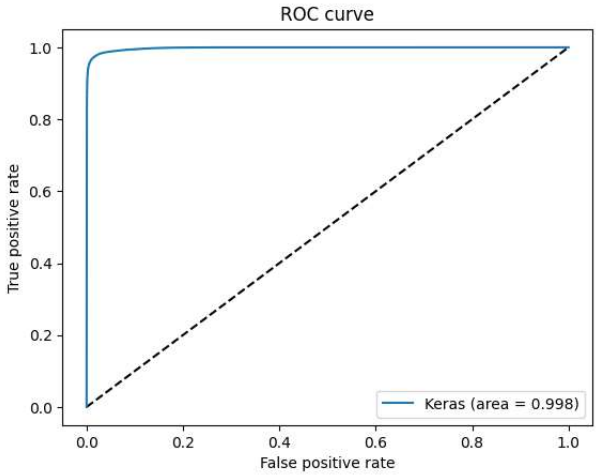


Appendix D: ROC curves of the binary models.

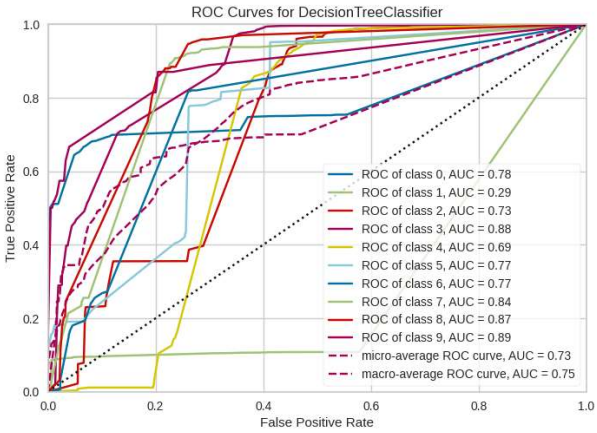


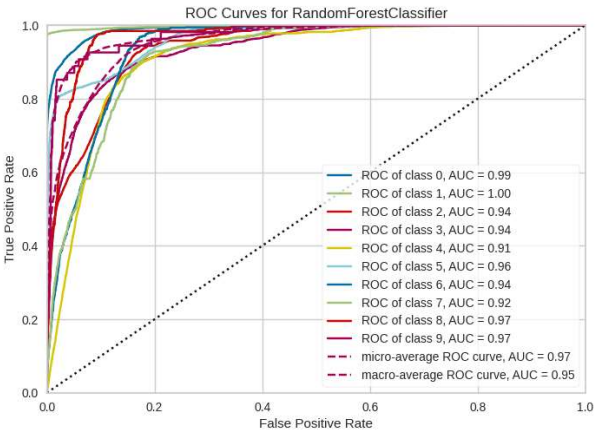


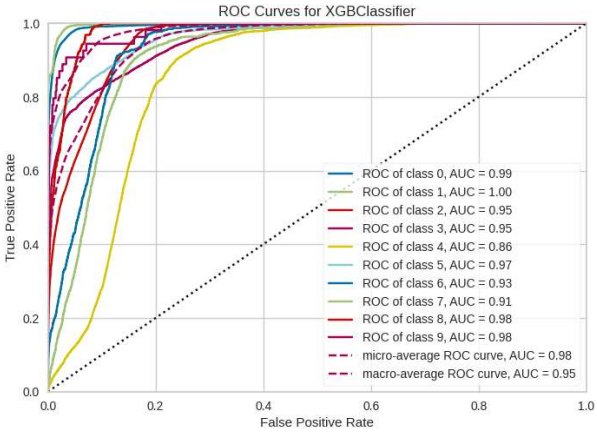


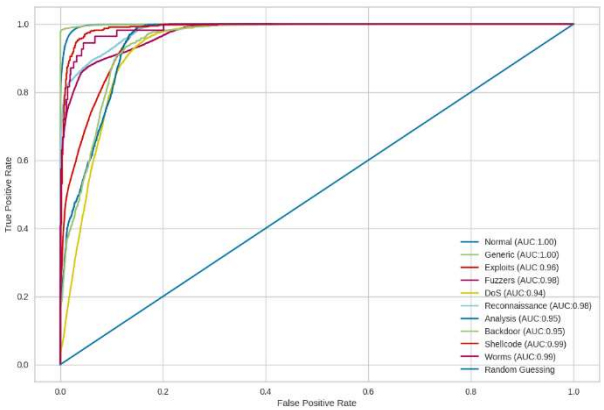


Appendix E: ROC curves of the multiclass models.

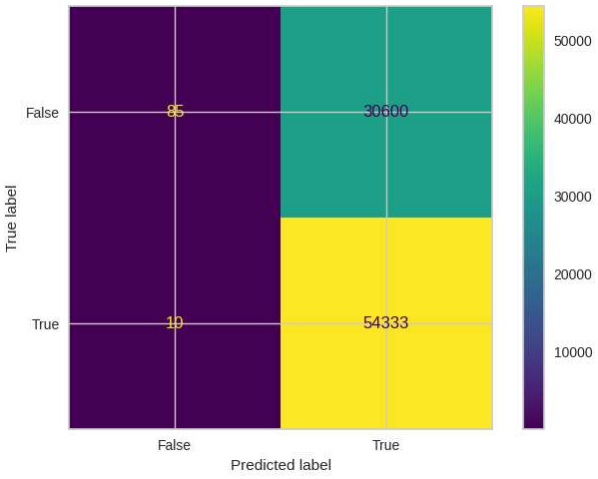


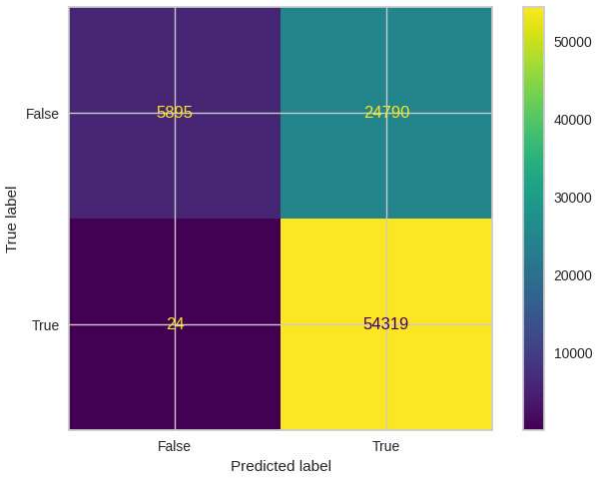


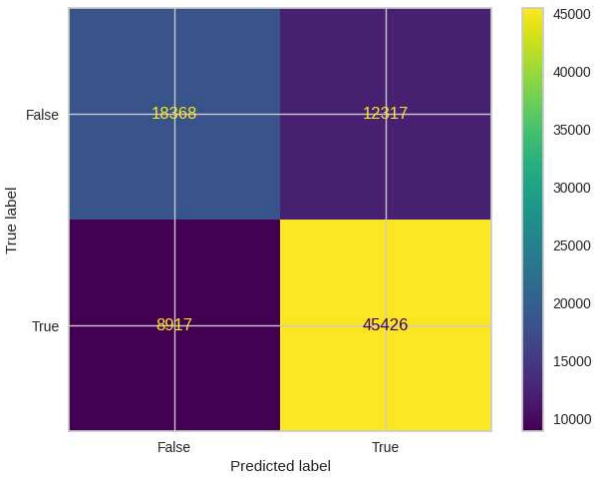


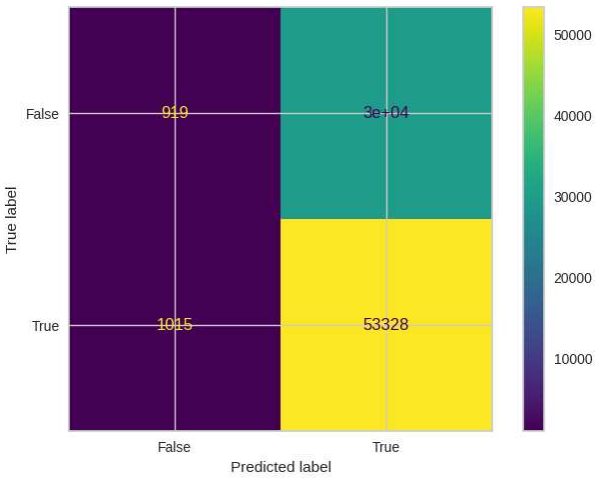


Appendix F: Confusion matrices of binary models with 90% of flipped labels (logistic regression top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

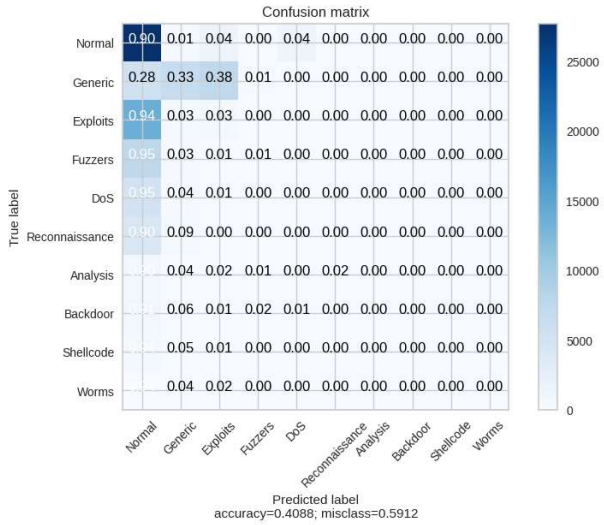


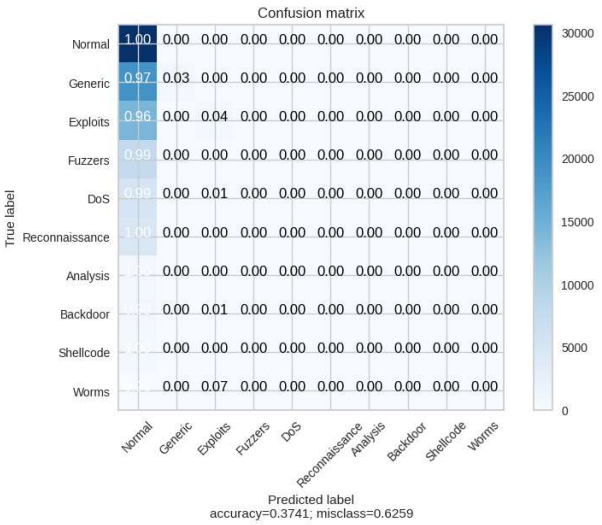


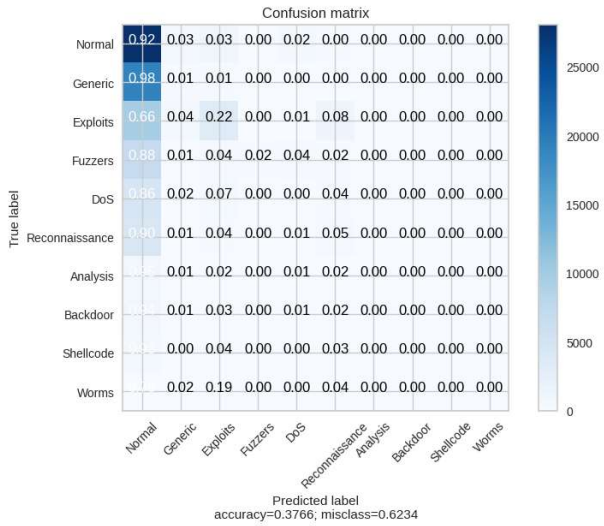


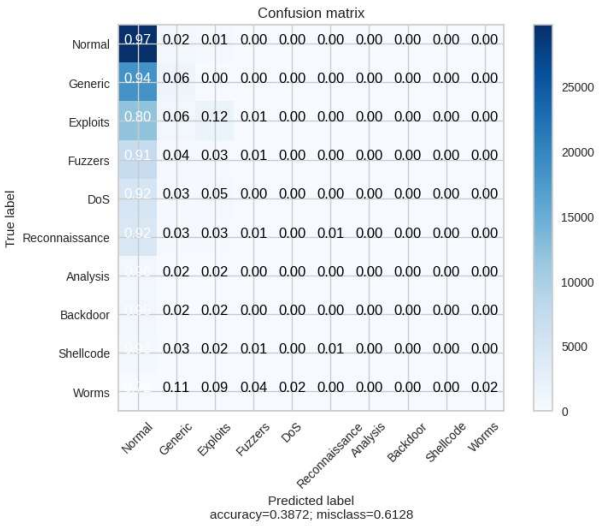


Appendix G: Confusion matrices of multiclass models with 90% of flipped labels (decision tree top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

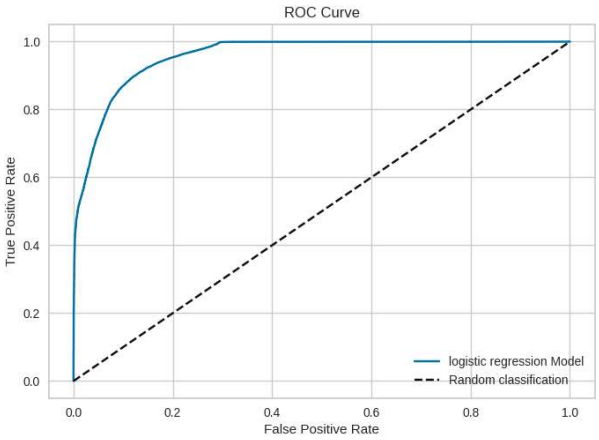


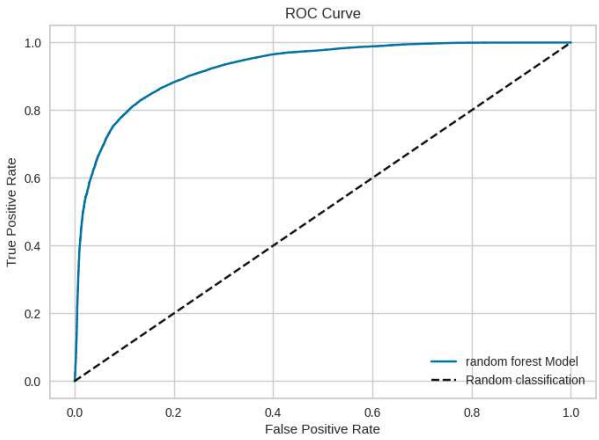


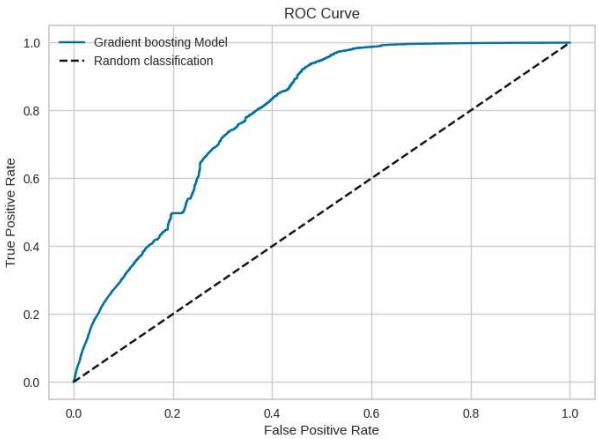


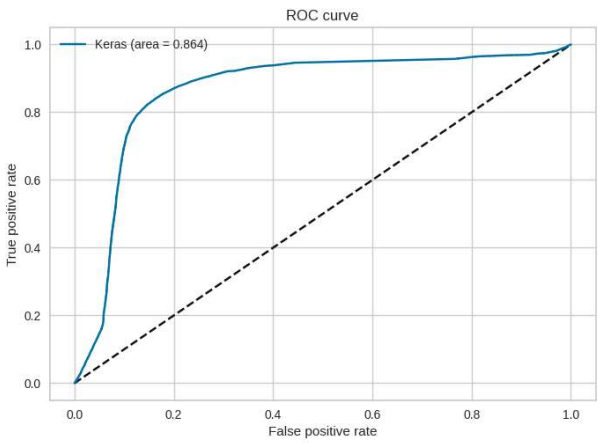


Appendix H: ROC curves of the binary models with 90% of flipped labels.

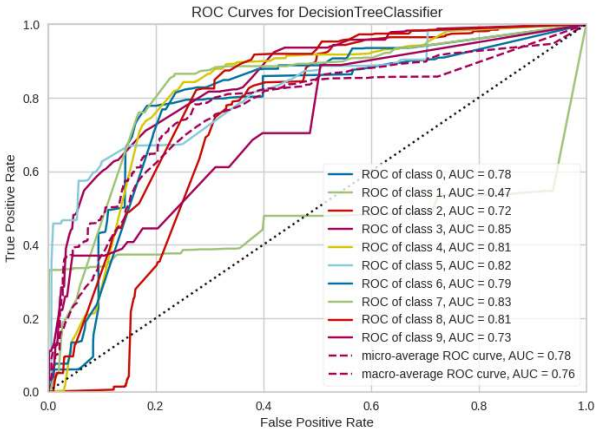


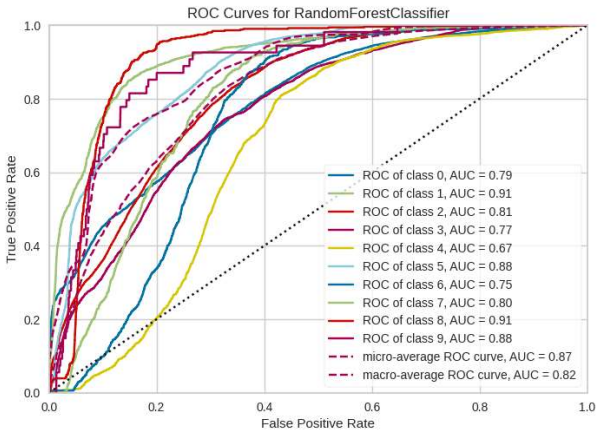


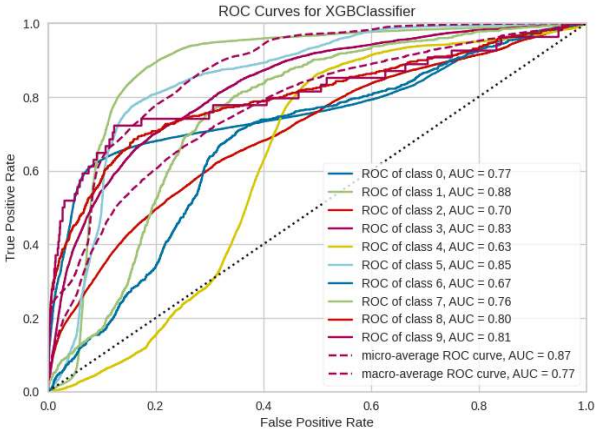


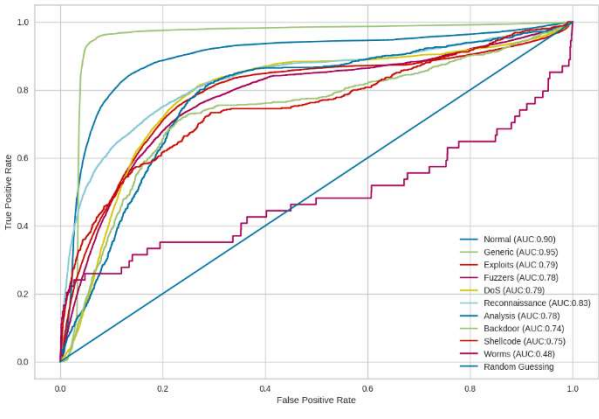


Appendix I: ROC curves of the multiclass models with 90% of flipped labels.

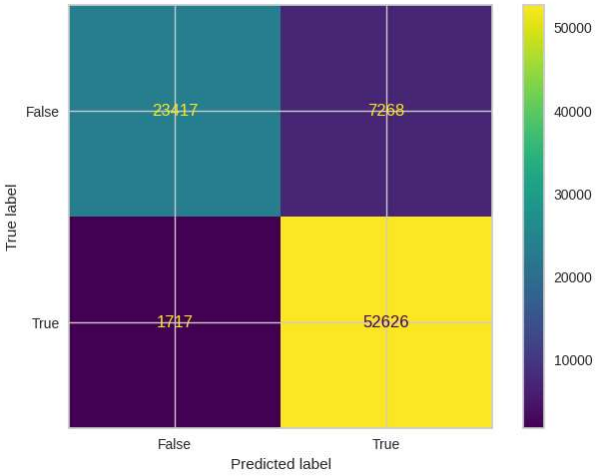


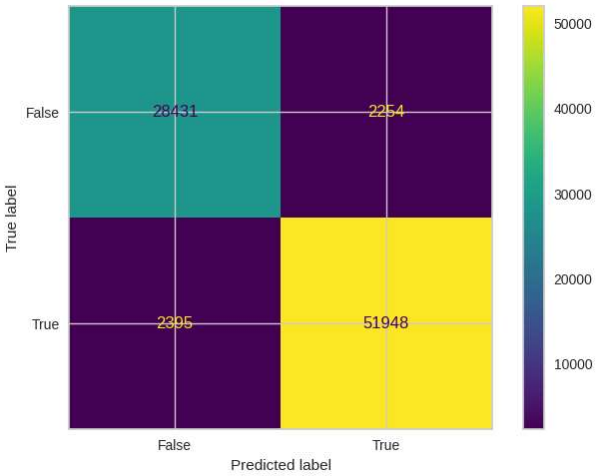


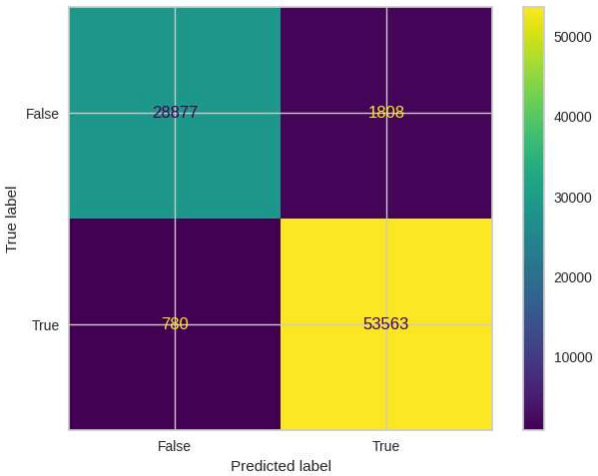


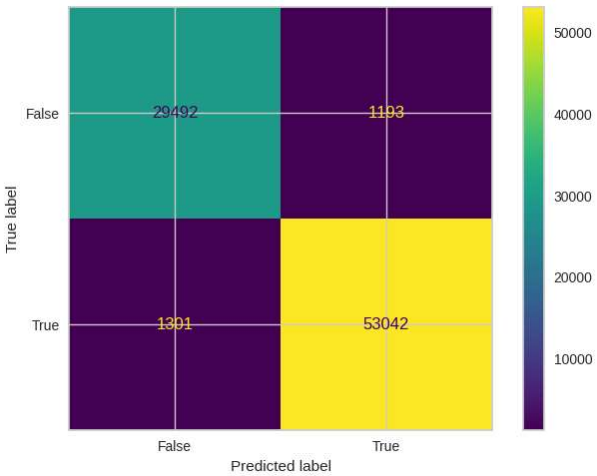


Appendix J: Confusion matrices of binary models with added random noise in range 0-6 (logistic regression top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

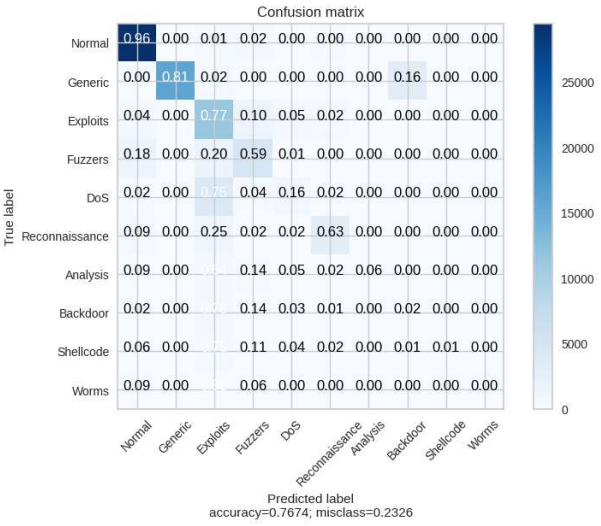


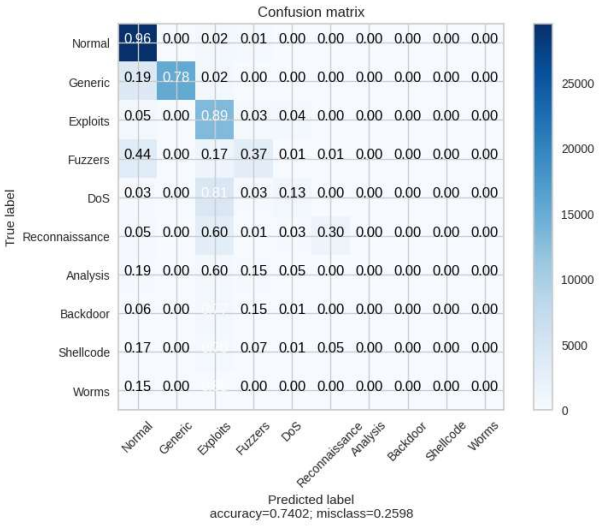


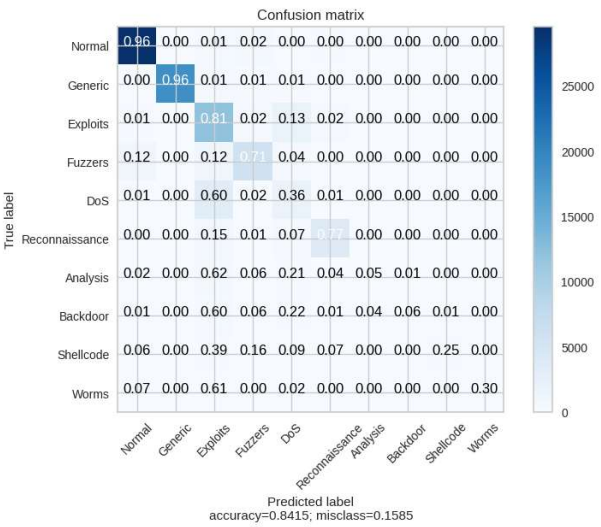


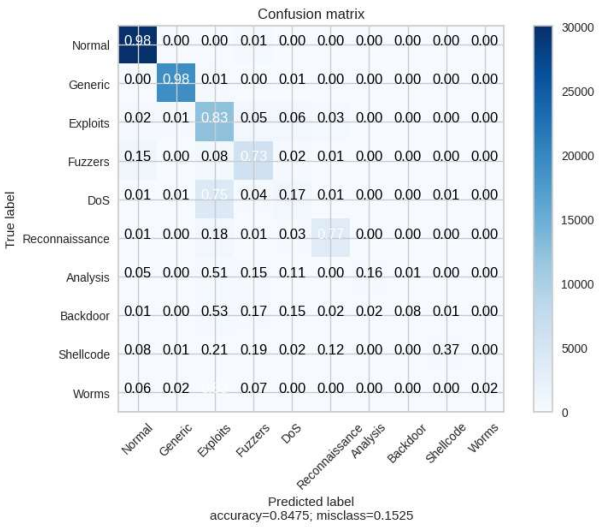


Appendix K: Confusion matrices of multiclass models with added random noise in range 0-6 (logistic regression top-left, random forest top-right, xgboost bottom-left, neural network bottom-right).

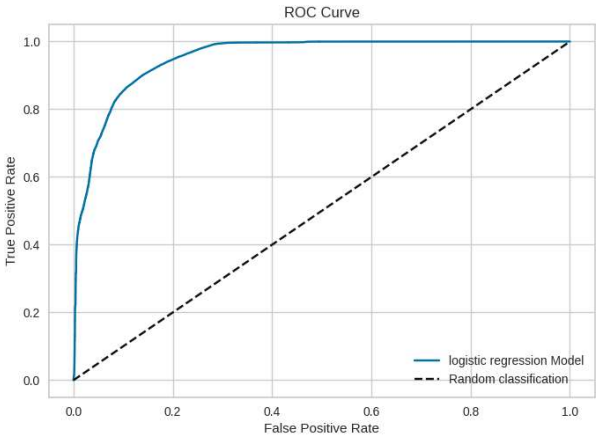


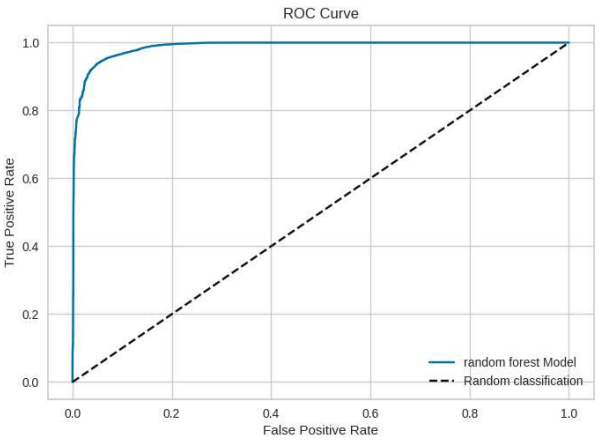


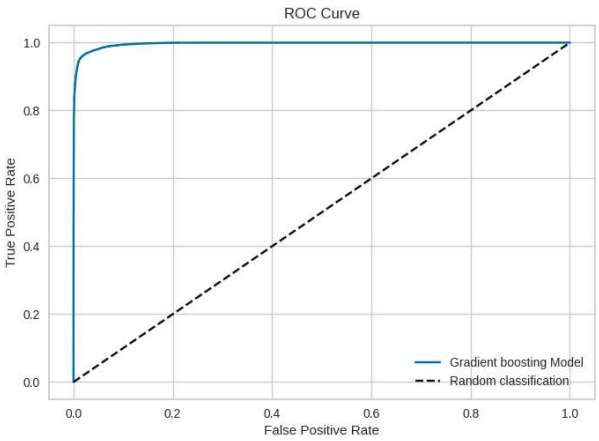


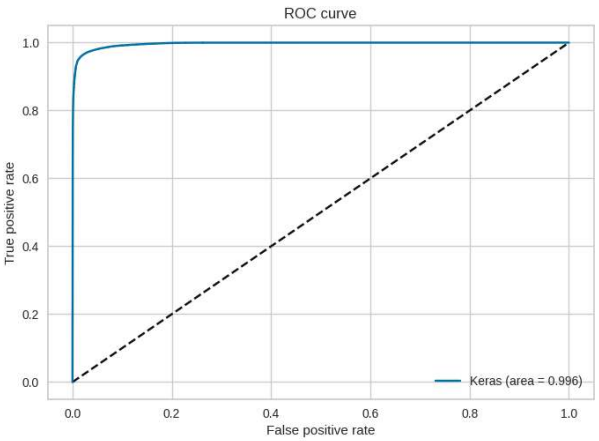


Appendix L: ROC curves of the binary models with added random noise in range 0-6.

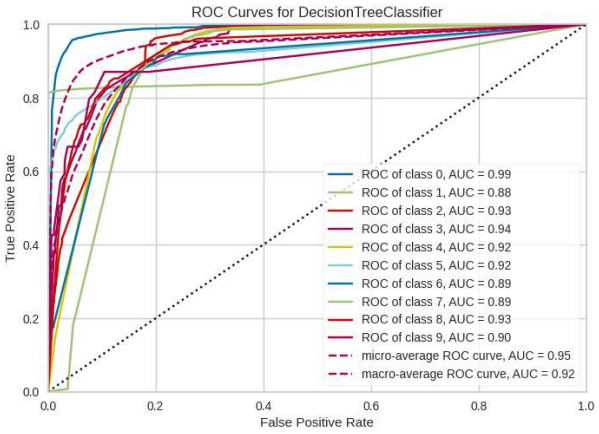


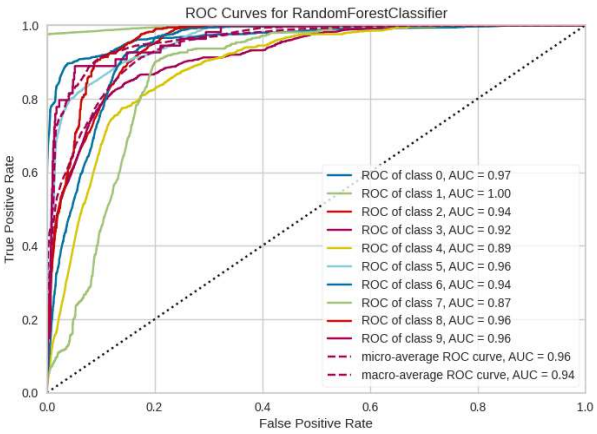


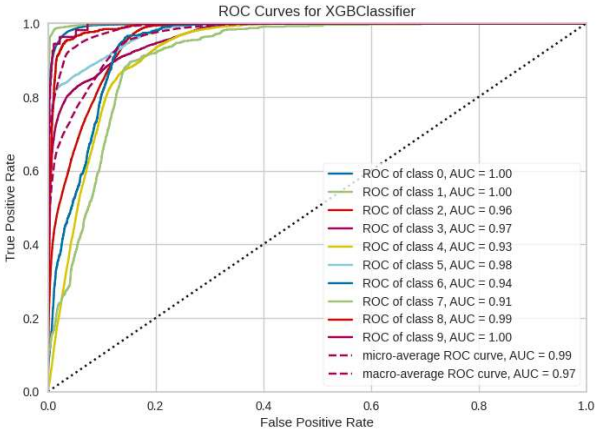


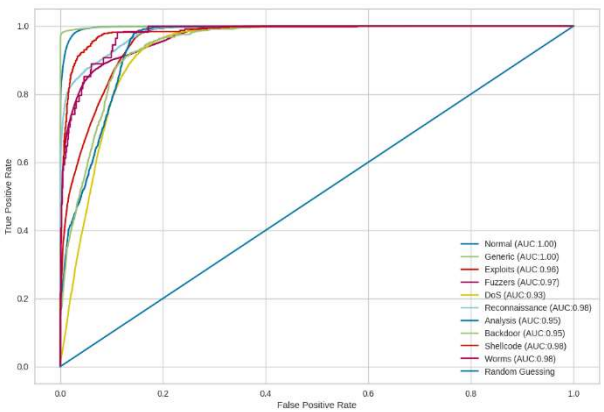


Appendix M: ROC curves of the multiclass models with added random noise in range 0-6.









9. References

Liao et al. (2013), “Intrusion detection system: A comprehensive review”, Journal of Network and Computer Applications, 36(1), 16-24, https://doi.org/10.1016/j.jnca.2012.09.004

Liu, Lang (2019), “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey”, Journal of Applied Sciences, 9(20), 43-96, https://doi.org/10.3390/app9204396

Gümüşbaş et al. (2020), “A Comprehensive Survey of Databases and Deep Learning Methods for Cybersecurity and Intrusion Detection Systems”, IEEE Systems Journal, 15(2) 1717-1731, 10.1109/JSYST.2020.2992966

Asharf et al. (2020), “A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions”, Electronics, 9(7), 1177, https://doi.org/10.3390/electronics9071177

Amouri et al. (2020), “A Machine Learning Based Intrusion Detection System for Mobile Internet of Things”, Sensors, 20(2), 461, https://doi.org/10.3390/s20020461

Pinto et al. (2023), “Survey on Intrusion Detection Systems Based on Machine Learning Techniques for the Protection of Critical Infrastructure”, Sensors, 23(5), 2415, https://doi.org/10.3390/s23052415

Moustafa et al. (2015) "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).", Military Communications and Information Systems Conference (MilCIS), IEEE, 10.1109/MilCIS.2015.7348942

Moustafa et al. (2016), "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset.", Information Security Journal: A Global Perspective, 25(1-3), 18-31, https://doi.org/10.1080/19393555.2015.1125974

Moustafa et al. (2017), "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks.", IEEE Transactions on Big Data, 5(4), 481-494, 10.1109/TBDATA.2017.2715166.

Moustafa et al. (2017) "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models.", Data Analytics and Decision Support for Cybersecurity, Springer, Cham, 127-156, https://doi.org/10.1007/978-3-319-59439-2_5.

Sarhan et al. (2020), “NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems”, Big Data Technologies and Applications: 10th EAI International Conference, Springer Nature, https://doi.org/10.48550/arXiv.2011.09144

Moustafa et al. (2018), “An Ensemble Intrusion Detection Technique based on proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things., IEEE Internet of Things Journal, 6(3), 4815-4830, 10.1109/JIOT.2018.2871719

Koroniotis et al. (2017), “Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques”, International Conference on Mobile Networks and Management, Mobile Networks and Management, Springer, Cham, 30–44, https://doi.org/10.1007/978-3-319-90775-8_3

Biggio et al. (2012), “Poisoning attacks against support vector machines”, Proceedings of the 29th International Conference on Machine Learning, https://doi.org/10.48550/arXiv.1206.6389

Ramirez et al. (2021), “Poisoning Attacks and Defenses on Artificial Intelligence: A Survey”, Cryptography and Security, https://doi.org/10.48550/arXiv.2202.10276

Kuzlu et al. (2020), “Role of Artificial Intelligence in the Internet of Things (IoT) Cybersecurity”, Discover Internet Things 1, Springer, 7, https://doi.org/10.1007/s43926-020-00001-4

Sun et al. (2015), “Data Poisoning Attacks on Federated Machine Learning”, IEEE Internet of Things Journal, 9(13), 11365-11375, 10.1109/JIOT.2021.3128646

Dunn et al. (2020), “Robustness Evaluations of Sustainable Machine Learning Models against Data Poisoning Attacks in the Internet of Things”, Sustainability, 12(6), 6434, https://doi.org/10.3390/su12166434

Stokes et al. (2021), “Preventing Machine Learning Poisoning Attacks Using Authentication and Provenance”, IEEE Military Communications Conference, 181-188, 10.1109/MILCOM52596.2021.9653139

Salvi et al. (2022), “Cyber-resilience of Critical Cyber Infrastructures: Integrating digital twins in the electric power ecosystem”, Computers & Security, 122, https://doi.org/10.1016/j.cose.2021.102507

Varghese et al. (2022) “Digital Twin-based Intrusion Detection for Industrial Control Systems,” IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, 611-617, 10.1109/PerComWorkshops53856.2022.9767492

Bhatia et al. (2021), “MSTREAM: Fast Anomaly Detection in Multi-Aspect Streams”, Machine Learning, https://doi.org/10.48550/arXiv.2009.08451

Siddiqui et al. (2019), “Detecting Cyber Attacks Using Anomaly Detection with Explanations and Expert Feedback”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2872-2876, 10.1109/ICASSP.2019.8683212

Ali et al. (2023), “Hybrid machine learning approach for construction cost estimation: an evaluation of extreme gradient boosting model”, Asian Journal of Civil Engineering, 24(7), 1-16, 10.1007/s42107-023-00651-z

Rosenfeld et al. (2020), “Certified Robustness to Label-Flipping Attacks via Randomized Smoothing”, Proceedings of Machine Learning Research, PMLR 119:8230-8241

Chollet, F. (2018), “Deep Learning with Python”, Manning

James, G. Witten, D. Hastie, T. (2017), “An Introduction to Statistical Learning with Applications in R”, Springer

(2023), “pandas documentation”, https://pandas.pydata.org/docs/

“scikit-learn machine learning in Python”, https://scikit-learn.org/stable/

“Spark By Examples | Learn Spark Tutorial with Examples”, https://sparkbyexamples.com/

“Keras: Deep Learning for humans” https://keras.io/

Constantin, L. (2021), “How data poisoning attacks corrupt machine learning models”, CSO, https://www.csoonline.com/article/570555/how-data-poisoning-attacks-corrupt-machine-learning-models.html

Monteagudo, J. “Artificial Intelligence & Machine Learning in Cyber Security”, https://cyberstartupobservatory.com/artificial-intelligence-cyber-security/

Gregory, J. (2021), “Data Poisoning: The Next Big Threat”, Security Intelligence, https://securityintelligence.com/articles/data-poisoning-big-threat/

Bhagat, V. (2022), “Artificial Intelligence(AI) in Cybersecurity: Future and Real Examples”, PixelCrayons, https://www.pixelcrayons.com/blog/artificial-intelligence-in-cyber-security-future-and-real-examples/#AI_can_be_manipulated

Ortner, A. (2020), "Top 10 Binary Classification Algorithms [a Beginner's Guide]", Towards Data Science, https://towardsdatascience.com/top-10-binary-classification-algorithms-a-beginners-guide-feeacbd7a3e2

Pujari, R. (2020), "Network Attack Detection and Classification Using Machine Learning Models Based on UNSW-NB15 Data-Set", Medium, https://i-rakshitpujari.medium.com/network-attack-detection-and-classification-using-machine-learning-models-based-on-unsw-nb15-a645bba73987

Tsui, K. (2018), "Perhaps the Simplest Introduction of Adversarial Examples Ever", Towards Data Science, https://towardsdatascience.com/perhaps-the-simplest-introduction-of-adversarial-examples-ever-c0839a759b8d

Brownlee, J. (2022), "How to Calculate Precision, Recall, F1, and More for Deep Learning Models", Machine Learning Mastery, https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/

(2022), "Learn Precision, Recall, and F1 Score of Multiclass Classification in Depth", Regenerative, https://regenerativetoday.com/learn-precision-recall-and-f1-score-of-multiclass-classification-in-depth/

Agrawal, S. K. (2021), "Evaluation Metrics for Classification Problem", Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

Shaffi, A. (2021), "Building a Confusion Matrix from Scratch", Medium, DataDrivenInvestor, https://medium.datadriveninvestor.com/building-a-confusion-matrix-from-scratch-85a8bfb97626

Godoy, D. (2018), "Understanding binary cross-entropy / log loss: a visual explanation" Towards Data Science, https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Koehrsen, W. (2017), "Random Forest Simple Explanation", Medium, https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

Gòmez, R. (2018), "Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names", Raùl Gòmez blog, https://gombru.github.io/2018/05/23/cross_entropy_loss/

Klintberg, A. (2017), “Explaining precision and recall”, Medium, https://medium.com/@klintcho/explaining-precision-and-recall-c770eb9c69e9

Verma, Y. (2021), “A Complete Understanding of Dense Layers in Neural Networks”, Mystery Vault, https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/