Part 2. Machine Learning

Martin Georgiev
1901560

1. Introduction

1.1 Background

In addition to the suggested secure software development practices, the investigator has conducted further research to enhance the organisation's Intrusion Detection System (from here on **IDS**). Currently, the company's IDS generates large amounts of raw data that the security team finds complex to analyse effectively. To address this issue, the investigator proposed a machine learning (from here on **ML**) classifier to bolster the network's resilience against attacks. By employing an ML algorithm to process the raw data and transform it into a more manageable format for the security team, the proposed solution is expected to increase threat detection efficiency and reduce false-positive alerts. Integrating ML with the IDS will lead to a more effective response to network attacks, resulting in a safer and more secure system.

1.2 ML implementation in IDS

Machine learning is a subset of artificial intelligence that relies on algorithms and data analysis to simulate a learning process with iterative improvement. By analysing data and identifying patterns, ML algorithms can learn from a provided training set to make predictions. Due to its predictive capabilities, ML's usage is continuously increasing across various industries. Four types of learning can be encountered when using ML – Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning.

Supervised Learning (Delua, 2021) uses labelled data to train the algorithm for pattern identification and prediction. When applied to the organisation's labelled data, it can detect potential network attacks based on network activity, leading to faster threat detection and response, and improved overall security posture. A diagram of the supervised learning process can be found in **Figure 1.1**.

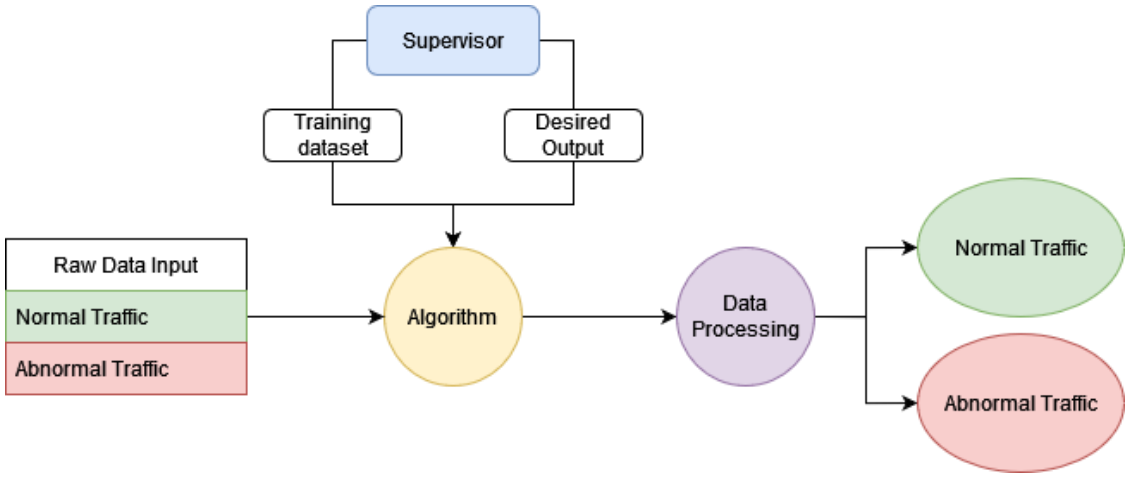

Figure 1.1 *– Supervised Learning Process.*

Unsupervised Learning (Delua, 2021) requires the algorithm to identify patterns within the data and categorise it as the information is not labelled. This learning method is often employed in cybersecurity solutions as it can easily and effectively adapt to security threats. In this case, the data will be categorised into the provided abnormal and normal traffic in code separate from the model, allowing it to flag the packets accordingly. This will also allow it to identify abnormal data for different variations of an exploit. An example diagram can be found in **Figure 1.2**. Semi-Supervised

is a hybrid version of the previously mentioned learning methods which is commonly used in document classification as it can be hard to find many labelled documents.

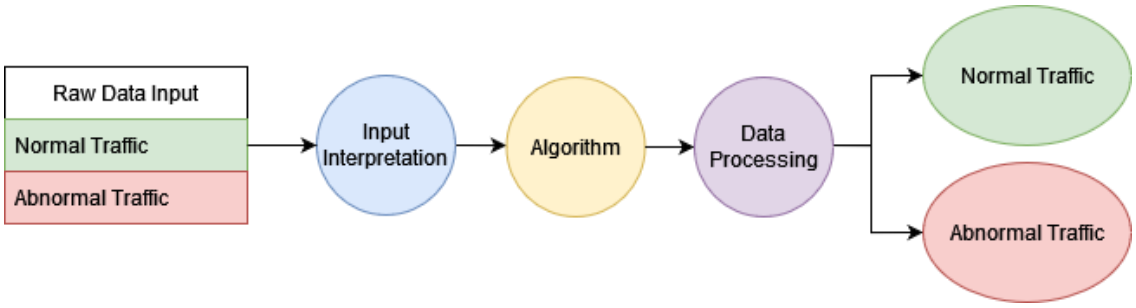

Figure 1.2 *– Unsupervised Learning Process.*

The final learning method, Reinforcement Learning (Bhatt, 2018), reinforces it either negatively or positively in response to specific behaviour. Such an agent can interpret its environment and learn through trial and error by taking specific actions. It uses delayed feedback to minimize the costs of an action. This learning method is commonly used in robotics, with a great example being overcoming obstacles such as mazes. An example diagram can be seen in **Figure 1.3**.

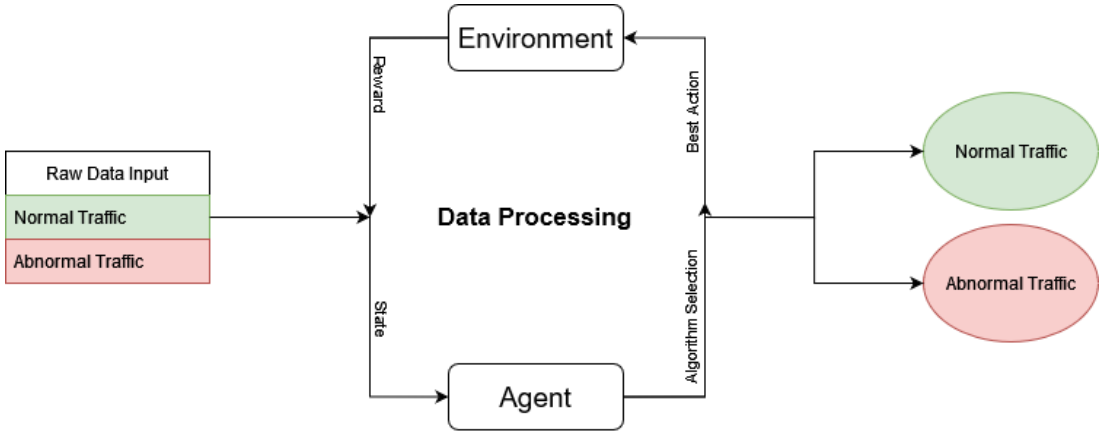

Figure 1.3 *– Reinforcement Learning Process.*

Considering the provided test dataset and the nature of the data (network packets with specific features), the investigator will focus on Supervised Learning algorithms – Multi-layer Perception (from here on MLP) Neural Networks and Random Forests.

1.3 Aims

Based on the provided data, the investigation will focus on the supervised and unsupervised learning methods. The investigation aims to design a classifier that can categorise the provided network packet data into two categories according to their types. Based on the provided network data, the researcher split them into normal and abnormal traffic. The packets contained various information regarding the service/protocols and attack category. The latter was used to create the beforementioned groups. The ten provided attack types can be seen in Table 1.1.

Abnormal	Normal
Fuzzers	Generic
DoS	Normal
Reconnaissance	Analysis
Backdoors	
Exploits	
Shellcode	
Worm	

Table 1.1 *– Attack categories in the provided network packet data.*

The report's objectives are as follows:

- ML Algorithm Discussion – In-depth discussion about two ML algorithms with their respective advantages/disadvantages, limitations and when they can be employed.
- Build Phases – In-depth description of the data pipeline and how the data will be processed, modelled, and analysed.
- Evaluation – Description of evaluation metrics appropriate to test the developed classifier.

2. Proposed Algorithms

2.1 MLP Neural Network

Multi-layer Perception (from here on MLP) is a type of Supervised Learning that uses an artificial neural network (Biswal, 2023). Neural networks are similar to the human brain and consist of different variations of neurons, also known as layers. The MLP algorithm employs multiple types of neurons to analyse data. The input layer receives the data for analysis, while the hidden layer processes the analysed data and learns from it. In a supervised classification system, each input neuron will be associated with a feature or a label (protocol, IP address, etc.) Finally, the output layer displays the synthesised results of the study. (**Figure 2.1**)

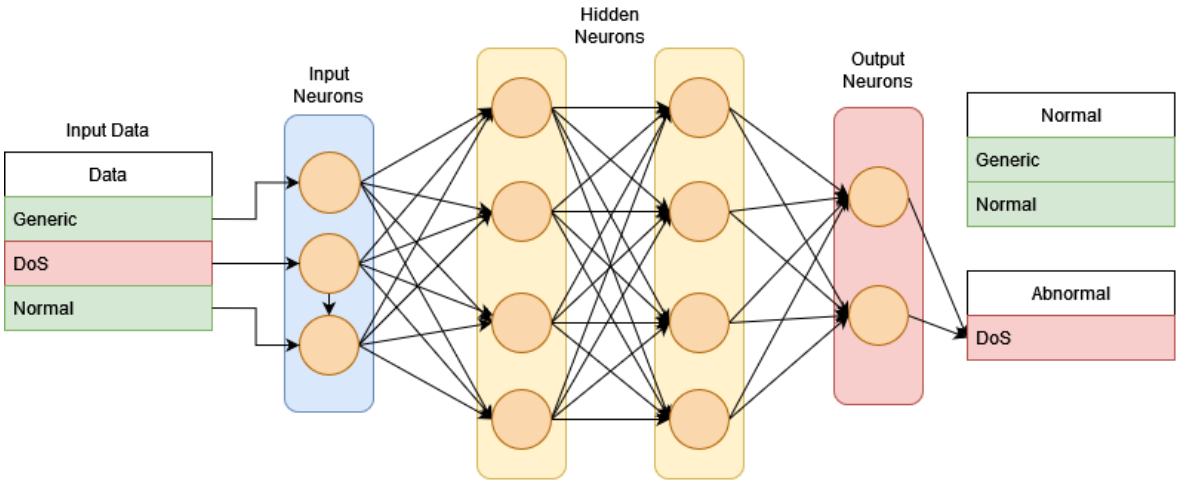

Figure 2.1 *– MLP example with part of the provided dataset.*

As previously discussed, the MLP algorithm employs hidden neurons to analyse input data. These neurons are organised into multiple layers that form a feature hierarchy. Each hidden layer processes the input of the previous layer, with the first hidden layer analysing data from the input neurons. This

process continues through the subsequent hidden layers until they reach the output neurons. Thus, the MLP algorithm incrementally analyses and learns from the input data until it produces a final output, making it a "feedforward" algorithm.

Neurons in a neural network are connected by "weights" (Bento, 2021) that determine the strength of connections between layers. Data is processed based on these weights, and occasionally a "bias" value (Alammar, 2016) is added before passing data to the next layer. Biases indicate the distance between the network's predictions and the intended value. Throughout learning, weights and biases adjust to produce the correct output. Low weight/bias values indicate minimal impact/assumptions, while high values indicate significant impact/assumptions.

MLP is promising for network analysis thanks to its ability to handle large amounts of complex unorganised data and output results in a user-friendly format. It accurately flags severe alerts because of neural network stability and fault tolerance. However, training time is a limitation, and neural networks lack detailed explanations for the output. It could be problematic in specific applications like IDS as it cannot explain why a packet was flagged (Rawat, 2022).

2.2 Random Forest

Another supervised learning method is the Random Forest algorithm (Yui, 2019). The forest consists of multiple decision trees, which are combined with a method called Bootstrap Aggregation to compare the output of each tree. The bootstrap method (Brownlee, 2018), otherwise known as "bagging", is utilised to ensure the predictions have the highest possible accuracy. Additionally, it decreases the chances of false positive/negative results, making them less likely to be encountered. The method uses the training set as an example and randomly selected packets from real-life data. To guarantee its seamless operation, the selected network data packets must be the same size as the dataset. Afterwards, the algorithm chooses random features from the bootstrap dataset to create a new training dataset for each decision tree. (**Figure 2.2**)

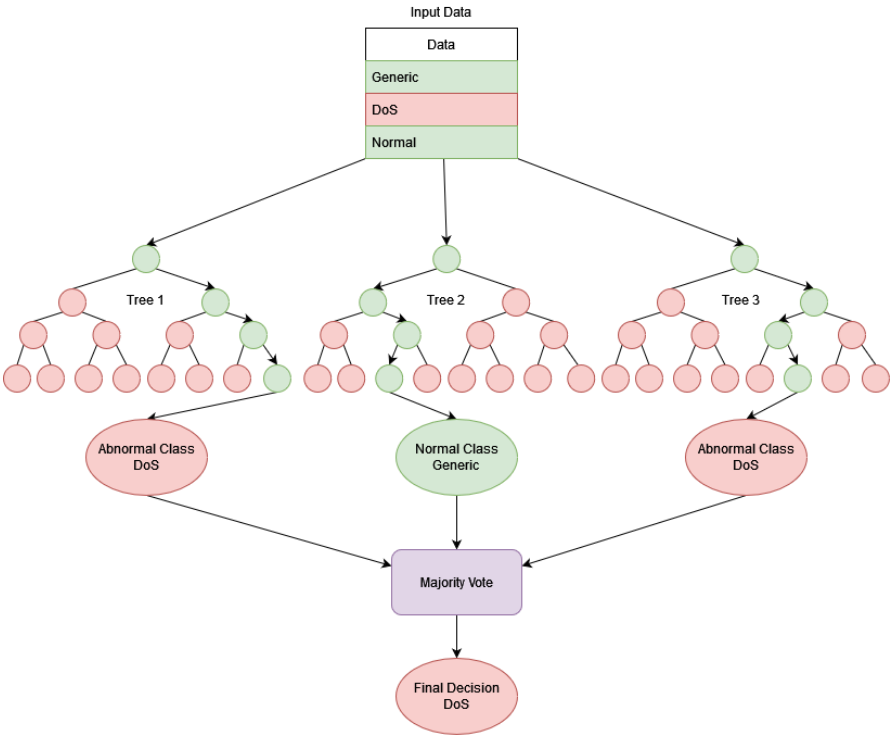

Figure 2.2 *– RF example with part of the provided dataset.*

Once each decision tree generates its output, the algorithm uses the new training dataset to compare the results and improve accuracy. For instance, if the algorithm employs ten decision trees and seven classify a packet as generic, it will be considered benign and will not be flagged for review. However, if six trees consider it abnormal, while five mark it as benign traffic, the packet will be flagged as malicious and forwarded to an analyst for further evaluation.

Comparable to the Multi-layer Perception algorithm, Random Forest is a suitable option for data analysis due to its accuracy and ability to handle large datasets. It is also immune to over-training and easy to use in an Intrusion Detection System. The algorithm can predict missing data to identify attacks, making it more resilient to modified versions of attacks and exploits. Regarding the disadvantages, RF requires significant computational power and hardware if many decision trees are used, which can slow down analysis or make it ineffective. Insufficient decision trees will result in lower accuracy while also making the forest prone to over-fitting (Brownlee, 2016) – performing well in the training data analysis while struggling with other provided data. Employment of enough trees and "pruning" (Lawrence, 2018) can counter the over-fitting issue and ensure higher accuracy (Donges, 2022). Finally, Random Forest cannot describe any relationships within the analysed data.

3. Classifier Build Phases

Comparing the analysed algorithms, the researcher recommends the random forest because of its flexibility, ability to predict missing data and the possibility to counter the outlined disadvantages. Additionally, RF can be employed within the IDS solution significantly faster than the Multi-layer Perception algorithm as it does not require extensive training. This makes it more efficient than the suggested neural network model. Furthermore, the provided dataset is both labelled and two-dimensional, making it suitable for a Supervised Learning model such as the Random Forest algorithm.

The following phases can be employed to effectively choose and develop the classifier:

- **Problem identification** – Identifying the use case is vital as it would establish the target, what data will be processed and what needs to be achieved.
- **Data acquisition** – How will the data be obtained and used within the model? The first data collection may be used as a training or testing dataset. As the researcher recommended RF, the data must be labelled within the code as the algorithm employs supervised learning.
- **Data Pre-processing** – The data must be standardised to ensure accurate predictions. Clean data (removed NULL or NaN values) (Kumar, 2018) will prevent the model from stopping before the end of the dataset.
- **Feature Selection** – Relevant features should be selected based on the use case (in this case IP addresses, ports, etc.) Appropriate feature engineering (scaling, proper features, multicollinearity) can improve performance. Multicollinearity is not expected in RF models, though correlated features may cause issues. Trees could utilise any of them as a predictor without preference over the other, which could result in inaccurate predictions. Appropriate feature selection is essential to mitigate this issue.
- **Model Selection** – The models will vary depending on the use case. The researcher has recommended Random Forest for the network traffic analysis.
- **Model Training** – Using the training and testing datasets to train the algorithm. Additionally, hyperparameter tuning could further increase the performance or make the predictions more accurate.

- **Additional Finetuning** – An adequate number of decision trees (which may vary depending on the use case and available hardware) is required for the prediction's accuracy and efficiency, and to avoid overfitting. If the number is low and overfitting occurs, tree pruning (removing non-collaborative parts) could improve the performance.
- **Results Evaluation** – The model can be assessed after the training phase via different evaluation metrics. Further information can be found in **Section 4 Classifier Evaluation**.
- **Classifier Deployment** – After the previous phases have been concluded, the model can be employed in the IDS to monitor for malicious traffic.
- **Maintenance and Monitoring of the Model** – The organisation must monitor the model's behaviour after deployment to ensure that the performance and predictions remain efficient and accurate. This can be achieved through **Model Evaluation**.

The model should be thoroughly tested with both training and mock datasets to confirm its efficiency and efficacy. Furthermore, the tests can be used to train the staff in how to analyse and communicate the model's output. The researcher suggests bar charts and line charts as they will aid with the prompt discussion and study of the results. Line charts can be used to identify the time frame of abnormal traffic. Bar charts, on the other hand, can be used to show each attack by category for a specific time scale – i.e., attacks on a daily, weekly, or monthly scale.

4. Classifier Evaluation

Performance evaluation is vital when creating a machine learning classifier. This phase should be carried out after the testing and training phases have been concluded. The researcher proposes two methods that can be employed to identify the prediction accuracy of the model – Confusion Matrix and Classification Report (Kreiger, 2020).

The Confusion Matrix (from here on CM) can show the accuracy of the prediction and what parts of the model would require additional training or changes. Scikit is a Python ML library that has a function for CM generation. The generated matrix displays each label, allowing for the identification of prediction inaccuracies. Scikit can also generate the Classification Report (from here on CR). The function considers multiple vital factors for a successful ML model – precision, recall, and F1 score (Harikrishnan 2019). The precision column shows the positive predictive value, while the recall column shows the sensitivity/positive rate. Finally, F1 takes both recall and precision to inspect the accuracy of each category. It is recommended to use CM and CR to evaluate the model's accuracy effectively.

Other evaluation methods such as comparing the baseline (initial prototype of the model and its results) with the final solution and an ROC graph can also be employed. The ROC graph will show the true and false positive results and compare them at different levels. Using them can show inconsistencies and what could be improved.

References

Delua, J. (2021). *Supervised vs. Unsupervised Learning: What's the Difference?* [online] Available at: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning [Accessed 2 Apr. 2023].

Bhatt, S. (2018). *Reinforcement Learning 101*. [online] Available at: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292 [Accessed 3 Apr. 2023].

Biswal, A. (2023). Top 10 Deep Learning Algorithms You Should Know in 2023. [online] Available at: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm [Accessed 5 Apr. 2023].

Bento, C. (2021). Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis. [online] Available at: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141 [Accessed 5 Apr. 2023].

Alammar, J. (2016). A Visual and Interactive Guide to the Basics of Neural Networks. [online] Available at: http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/ [Accessed 5 Apr. 2023].

Rawat, S. (2022). *Advantages and Disadvantages of Neural Networks*. [online] Available at: https://www.analyticssteps.com/blogs/advantages-and-disadvantages-neural-networks [Accessed 5 Apr. 2023].

Yui, T. (2019). *Understanding Random Forests - How the Algorithm Works and Why it Is So Effective*. [online] Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 [Accessed 6 Apr. 2023].

Brownlee, J. (2018). *A Gentle Introduction to the Bootstrap Method*. [online] Available at: https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/ [Accessed 6 Apr. 2023].

Brownlee, J. (2016). *Overfitting and Underfitting With Machine Learning Algorithms*. [online] Available at: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ [Accessed 6 Apr. 2023].

Lawrence, B. (2018). *Decision Trees — Pruning*. [online] Available at: https://towardsdatascience.com/decision-trees-pruning-4241cc266fef [Accessed 6 Apr. 2023].

Donges, N. (2022). *Random Forest: A Complete Guide for Machine Learning*. [online] Available at: https://builtin.com/data-science/random-forest-algorithm#feature [Accessed 6 Apr. 2023].

Kumar, D. (2018). *Introduction to Data Preprocessing in Machine Learning*. [online] Available at: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d [Accessed 9 Apr. 2023].

Kreiger, J. (2020). *Evaluating a Random Forest Model*. [online] Available at: https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56 [Accessed 17 Apr. 2023].

Harikrishnan, N. (2019). *Confusion Matrix, Accuracy, Precision, Recall, F1 Score*. [online] Available at: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd [Accessed 17 Apr. 2023].