



**Title: Artificially Intelligent SIEM for Targeted threat detection and analysis**

**By: Adam Cooper**

***Course: Cyber Security and Forensic Computing***

**Project code: PJR40 *PJE40***

**Supervisor: Hamidreza Khaleghzadeh**

**05/2024**

**Word count: 9,805**

SS

<input checked="" type="checkbox"/>	I give permission for my project to be published in the University library and/or be made available to other students as examples of previous work. (optional).
<input checked="" type="checkbox"/>	I confirm that I have read and understood the University Rules in respect of plagiarism and student misconduct.
<input checked="" type="checkbox"/>	I declare that this work is entirely my own. Each quotation or contribution cited from other work is fully referenced.

Date: \_03/05/2022\_

# Abstract

A prevalent issue within cyber security is the continuous fight to reduce false positive alerts, these mainly propose the largest threat to those investigating the alerts, wasting resources and time investigating that would be better spent elsewhere.

This project takes a look into the possibility of applying federated learning, specifically logistic regression and FedAVG to reduce false positives within a SOC environment. This was carried out using a combination of scikit-learn, flower federated learning and UNSW-NB15 dataset.

Software, hardware and health challenges led the project to deviate, however, the logistic regression model performed moderately well. Despite this, a high loss percentage potentially due to the model originally being overfitted did impact the performance.

This project brings to light the possibility for the use of federated learning within security applications; specifically, those revolving around threat detection and analysis. The area will need more development, with implementation of a model such as QRF or NNR, usage of TensorFlow and GPU acceleration to ensure the best possible outcome.

# Table of Contents

Abstract.....	3
Acknowledgements.....	7
1 Introduction .....	8
1.1 Background.....	8
1.2 Project Approach & Context .....	9
1.3 Significance of Project .....	9
1.4 Research Questions .....	10
1.5 Aims / Objectives.....	10
1.6 Project Constraints .....	10
1.7 Legal, Ethical and Professional Issues .....	10
1.8 Project Management .....	11
2 Literature Review .....	12
2.1 Introduction.....	12
2.2 Federated and classic machine learning differences.....	12
2.2.1 Basic Machine learning .....	12
2.2.2 Federated Machine learning .....	12
2.2.3 Data Privacy.....	13
2.2.4 Data Distribution .....	13
2.2.5 Continual Learning .....	13
2.2.6 Dataset Aggregation.....	13
2.1 Existing research .....	13
2.2 Algorithms & Techniques.....	15
2.2.1 Unsupervised Machine Learning (UML).....	15
2.2.2 Supervised Machine Learning (SML).....	16
2.2.3 Ensemble Machine Learning (EML).....	17
2.3 Gaps in research .....	18
2.4 Why Federated learning? .....	18
3 Research Methodology & Design .....	19
3.1 - Introduction.....	19
3.2 - Research Methodology.....	19
3.2.1 – Data procurement and importing .....	19
3.2.2 – Data preprocessing.....	20

3.2.3	Data Cleaning .....	20
3.2.4	Encoding.....	20
3.2.5	Normalisation.....	20
3.2.6	Results .....	20
3.3	Virtual Machines.....	21
3.3.1	– Ethical Considerations .....	21
3.4	– Search Methodology.....	21
3.5	– Model Algorithm Requirements .....	22
3.6	– Supervised, Unsupervised or Ensemble Learning .....	22
3.7	– Virtual Environment.....	22
3.8	– Hardware Specifications .....	23
3.9	– Software Specification .....	23
3.10	– Conclusion .....	23
4	– Results & Discussion.....	24
4.1	Windows server setup .....	24
4.2	Vulnerable Machine Setup .....	24
4.3	Ubuntu VM Creation.....	24
4.4	CentOS VM Creation .....	24
4.5	Kali Setup .....	25
4.6	SIEM Implementation .....	25
4.7	Quantile regression implementation.....	25
4.8	<b>Logistic Regression Model</b> .....	25
4.9	Initial approach.....	26
4.10	– Importing Data.....	26
4.10.1	– Dataset limitations .....	26
4.11	Pre-Processing .....	27
4.12	Cleaning .....	27
4.13	Entries with null values.....	27
4.14	Entries with no unique values .....	27
4.15	Redundant Value Check.....	28
4.16	Removal of null and duplicates.....	29
4.17	One hot encoding .....	29
4.18	What is it?.....	29
4.19	Implementation .....	29
4.20	Sorting labels and features .....	31

4.21	Normalisation .....	32
4.22	Revised Code .....	32
4.23	Results.....	33
4.24	Training .....	33
4.25	Testing.....	34
4.26	Conclusion, Reflection & Future Work .....	34
4.27	Setbacks & Challenges .....	35
4.27.1	System Issues .....	35
4.27.2	Networking Constraints & Documentation Challenges .....	35
4.27.3	Health Issues .....	36
4.28	Successes & Research progress .....	36
4.28.1	Resilience .....	36
4.29	Research questions and Future improvement .....	36
4.30	Personal Development .....	37
5	References .....	38
6	Appendices.....	41
6.1	Appendix A: PID .....	41
1.	Basic details.....	41
2.	Degree suitability .....	41
3.	The project environment and problem to be solved .....	42
4.	Project aim and objectives.....	43
5.	Project constraints .....	43
6.	Facilities and resources .....	43
7.	Log of risks.....	44
8.	Project deliverables.....	45
9.	Project approach .....	45
10.	Project Tasks and Timescales .....	46
11.	Supervisor Meetings .....	47
12.	Legal, ethical, professional, social issues .....	48
6.2	Appendix B: Gantt chart .....	0
6.3	Appendix C: Code .....	0
6.4	Appendix D: Ethics Form.....	1

# Acknowledgements

The author would like to thank his family and friends for their never-ending love and support throughout the duration of the project despite the issues faced throughout. This project is dedicated to Amanda and Jeffery Cooper, Adam's parents and both Louisa Painter and Richard Cooper, Adam's siblings.

# 1 Introduction

## 1.1 Background

(Cyber Security Breaches Survey 2023 - GOV.UK, 2023) Cyber attacks in recent years have become more prominent with 32% of businesses reported of having a data breach or an attack between 2022 and 2023. This rises considerably to 59% when looking at medium to large organisations and again even further to 69% when looking at large businesses with high-income, for example trading companies such as stocks & bonds, banking, finance or quantitative trading.

Also documented in the same report is the importance of security risks and steps taken to prevent them by these different sized organisations. Security priority is 12% less in smaller businesses where security infrastructure is non-existent. In comparison to 2021 the use of password policies, network firewalls, restriction of admin rights and application of updates within a respectable timeframe have all declined by 10% +/- 2%; all are common areas which threat actors exploit first.

(Committee on National Security Systems (CNSS) Glossary, 2022) Many companies attempt to solve the problems noted above regarding security issues by implementing intrusion detection (IDS) and Security information and event management (SIEM) systems within their network at various points to prevent or detect any harmful or unexpected activity or 'intrusion'. Intrusion being classed as any security event, singular or multiple, which is network or host-based activity and the threat actor "attempts to gain access to a system or system resource without having authorisation".

A big flaw with both an IDS and SIEM is that the systems rely purely on rules created by humans looking at specific features for an output. Although this allows you to pinpoint the criteria you would like to match on, it also opens your infrastructure up to separate issues:

- Higher false positives (FP) – Receiving an alert for malicious activity that isn't.
- Higher false negatives (FN) – Malicious activity labelled non-malicious.
- Misconfigured rules – Potential to either create noise, block important services or allow malicious activity permanently.
- Alert noise – E.g. many alerts could meet the following alert criteria 'match: cmd: "kubectl config set-context --current", contain: servername:"linuxeng-\*"'

Machine learning implemented within Extended Detection and Response (XDR) systems have proven to reduce the number of FP and FN (Pissanidis & Demertzis, 2024) in alerting along with reducing alert noise relating to poor rule creation while increasing the accuracy over all of results due to the wider range of data considered in each decision. Federated or distributed learning enhances machine learning through re-defining analysis to provide in depth analysis of datasets and predictions from trained models calculated by multiple clients.

These subjects will be looked at closer during the literature review stage in line with the research questions for this paper: Can AI effectively and accurately reduce alert noise from false positive alerting and similar previous activity whilst providing a more granular overview to alerting by implementing federated learning into a SIEM?, Can we utilise federated learning to effectively reduce false positive alerting? and Is federated learning an option for threat detection?

This will be accomplished by the following:

- Create a federated learning algorithm for intrusion detection.



- Train and test a model using data collected from real scenarios.
- Finalise and review results comparing to current machine learning solutions.

Overall, the literature review will look into federated learning along with any previous research and work carried out on false positives within current machine learning approaches and any relevant algorithms to find an approach best suited for this project.

## 1.2 Project Approach & Context

Project management and delivery will use Agile and be maintained through Jira. Secondary research will be carried out in the form of a literature review covering machine learning for anomaly detection and artificially intelligent SIEMs. As the subject of AI is relatively new to me, I will also be learning this throughout the duration of the project by referencing official documentation, i.e. scikit-learn.

In addition, I will also:

- Complete a review of machine learning models with what would be best suited for the project i.e., researching different algorithms for anomaly detection such as KNN and Decision trees.
- Build and develop a virtual environment, similar to a SOC for gathering attack data logs to simulate a real-life scenario for threat intelligence.
- Create machine learning model(s) and deploy against test data (aggregated datasets) and logs gathered from test environment.
- Implement system and model validation to confirm the model detected threats as expected and did not need altering.
- Evaluate both system and model to ensure they successfully fulfilled their requirements and explore potential improvements.

## 1.3 Significance of Project

The main audience of this project/report is cyber security analysts and those within security operation centres. These will be the main users of SIEM tools alongside threat detection and analysis tools that will be used daily to prevent attacks on the estate.

This research should provide a closer look into AI tooling for SIEM's, specifically targeting threat detection and analysis to prevent noise, false positives and increase alerting infrastructure by providing more in-depth alerting. Currently SIEM's such as siembol and Apache Metron, which do not have any AI implemented, lack this, and face the following issues:

- Alert Noise – Due to poorly setup or bad alerting.
- False positives – Poor alerting based on rules and no other activity.
- Alerts allowlisted not on a granular level – Activity excepted based on little common factors.
- Lack of consistency (i.e. alerts aren't updated in time) – Rules must be manually created and maintained.

Currently, AI in Cyber is still a niche sector with very few companies engaging in and implementing AI into their products. Similarly, and even more niche, AI used for threat detection and analysis with Artificially intelligent SIEM's are only being provided by a handful of companies these companies

being some of the most recognisable on the market: Elastic, Darktrace, Exabeam and Solarwinds to name a few.

#### 1.4 Research Questions

Can AI effectively and accurately reduce alert noise from false positive alerting and similar previous activity whilst providing a more granular overview to alerting by implementing machine learning into a SIEM?

Can we utilise federated learning to effectively reduce false positive alerting?

Is federated learning an option for threat detection?

#### 1.5 Aims / Objectives

**Aim:** To successfully implement AI into a SIEM for increased efficiency of alerting and reducing alert noise

**Objectives:**

- Complete a review of previous works on AI SIEM implementations and machine learning for anomaly detection.
- Create virtual SOC environment to produce realistic results.
- Create Machine learning model.
- Train and test against previous datasets.
- Implement machine learning into SIEM to test against logs from SOC Environment
- Validate and Evaluation of results.
- Submit a project which has had an impact on the research aim and answered research questions.

#### 1.6 Project Constraints

- Project timescale.
- Possibility of GDPR compliance issues.
- BCS code of conduct.
- University code of conduct.
- Personal hardware limitations.
- Networking limitations.

#### 1.7 Legal, Ethical and Professional Issues

Any work carried out within this project is my own work and tools that are used throughout will be opensource with their sources stated and compliance to their conducts along with GDPR to ensure that no personally identifiable (sensitive) information is being collected and/or processed, and compliance with the university code of conduct and BCS code of conduct. Any data collected is to be destroyed at the end of the project.

There will be no human participants as part of this project, the network will be open to the public and therefore adversaries can engage and attack the network at their own will. The network will be separated from any other network, running in a demilitarized zone and all systems are run from within virtual machines.

There will be no financial inducements or risks to physical or environmental features, historical or cultural heritage. The project does not involve animals and will not be harmful to any third parties nor can it affect UK security.

## 1.8 Project Management

Jira

## 2 Literature Review

### 2.1 Introduction

This chapter will review the surrounding literature of that highlighted in the [introduction](#). Beginning with a deeper look into federated learning and a comparison to machine learning. Following on, algorithms frequently used for threat and anomaly detection will be researched to identify one best suitable for this project; these will also be split into their respective machine learning techniques.

### 2.2 Federated and classic machine learning differences

Documented in (Joshi, 2022), there are four key differences between these two entities but firstly, how exactly does federated learning work in comparison to basic machine learning?

#### 2.2.1 Basic Machine learning

Standard machine learning comprises of predictions made by algorithms based on some sort of data which it will train on and through this data performance will increase providing more effective, accurate predictions(Sodhi et al., 1354). Machine learning level of AI is similar to learning a language for a person, for example, someone may look at a word that is not in their native language, recognise it then think of what the translation for that specific word is except AI does this much faster by having a direct reference to the word which helps to provide an instant response.

#### 2.2.2 Federated Machine learning

Federated learning uses basic machine learning methods meaning the same knowledge of models and techniques are still applied, however, this implements clients and servers. Each server will have multiple clients connected to it at runtime(Wen et al., 2023).

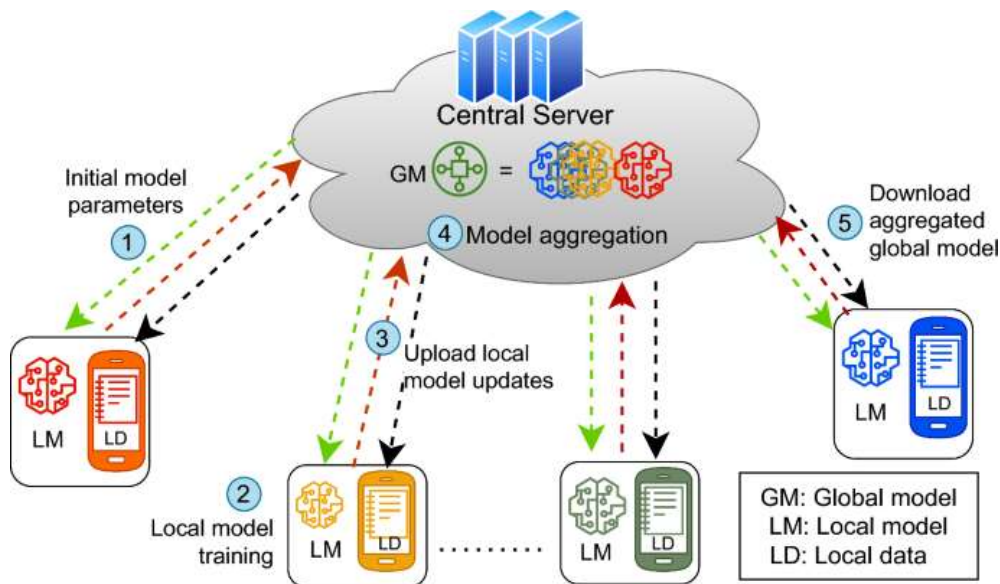


Figure 1 – Diagram outlining the federated network (Shanmugarasa et al., 2023)

Figure 1 shows presents a good representation of this. The server holds a general model based on a training dataset provided, this model is then used by each of the clients connected to it. The clients then train and test against a local dataset using the model parameters received from the FL server and once all round of

learning have completed, updates of the local model that has been trained are relayed back to the server where results from each of the clients are aggregated, the GM is updated and the process can repeat.

### 2.2.3 Data Privacy

With the introduction of federated learning, data privacy in comparison to basic machine learning methods increased. The ability to train a model and retain privacy of data was previously not possible as information was required to be shared with the central server (Liu et al., 2023). Federated learning creates an airgap between the data stored on clients and data stored on the central server, no data will leave the client, only the results of the federated learning.

### 2.2.4 Data Distribution

Basic machine learning distributes data to participants identically whereas this is not the case with federated learning due to the dataset being localised to the clients. Some datasets may be larger or smaller than others which would skew the result sent back to the server to train the global model. To counter this FedAVG is commonly used (Konečn et al., 2016) and will also be used in this project, the server averages results received from each client then adds the averaged results to the global model before sending the new parameters to the clients.

### 2.2.5 Continual Learning

In contrast to privacy and distribution, continual learning is where traditional machine learning shines in comparison to federated learning. Where clients do not have access to an overarching dataset, supported in (Kairouz et al., 2019), compared to traditional methods.

### 2.2.6 Dataset Aggregation

Traditional machine learning originally aggregates client data on the server as highlighted when discussing the privacy differences between ML and FL. Rather than aggregating data, when using federated learning techniques aggregation only takes place on results received from the clients and not the data they used to generate said results.

## 2.1 Existing research

Many research papers and projects highlighting de-centralised intrusion detection previously have mainly worked and focused research on models, heavily emphasising on workings behind the model itself and parameters transferred between client and server. As discussed in (Sarhan et al., 2023) Another common occurrence between these multiple research papers and many other projects seems to be any model training was carried out utilising a singular dataset usually an 80:20 split for train-to-test data with to the best of my knowledge, only several using more than one dataset or attempting to make the environment used as accurate to a real scenario as possible.

It's also important to note that although different, most research utilising decentralised machine learning focus around IoT devices, specifically those that fall within the smart home devices category and not desktop computers or servers which are equally as vulnerable in the threat landscape.

(Campos et al., 2022) explores federated learning for intrusion detection within an IOT setting. It's explained that the attack surface with recent technological advances is significantly advancing, resulting in devices becoming more susceptible to sophisticated attacks. A good and well-known example of this is the Mirai attack and its subsequent variants (Augusto Remillano II & Jemimah Molina, 2020) which use Telnet (ports 23&2323) and SSH (ports 22/custom) to brute force IoT devices. The comparison to traditional machine learning was carried out against a series of existing works many using outdated datasets such as KDDCUP99 produced 24 years ago and its successor the NSL-KDD produced 2009 with some comparison again more recent such as bot-iot but again in a emulated lab environment by the same producers as ToN-IoT used here.

(Bukhari et al., 2024) proposes a federated learning stacked convolutional neural network and bidirectional long, short term memory (FL-SCNN-Bi-LSTM) model for analysis of intrusion patterns utilising deep learning techniques such as special data collection and temporal relationship capture. Principal component analysis (PCA) was also used to provide better feature selection when training and testing against the CIC-IDS2017 dataset which features denial of service attacks.

**Table 5**  
Comparative Analysis of different techniques with proposed model.

Dataset	Base proposed technique	Proposed model evaluated metrics (%) without FL	Proposed model evaluated metrics (%) with FL
WSN-DS (Dataset 1)	SCNN-Bi-LSTM	Accuracy = 0.995 F1-Score = 0.994 Precision = 0.995 Recall = 0.994	Accuracy = 0.997 F1-Score = 0.996 Precision = 0.998 Recall = 0.996
CIC-IDS2017 (Dataset 2)	SCNN-Bi-LSTM	Accuracy = 0.9990 F1-Score = 0.9990 Precision = 0.9990 Recall = 0.9991	Accuracy = 0.9993 F1-Score = 0.9993 Precision = 0.9993 Recall = 0.9992

Figure 2 - SCNN-Bi-LSTM research findings (Bukhari et al., 2024)

The results from this research shown in Figure 2 display marginally better performance over non federated machine learning approach using the same technique.

In (Evangelou & Adams, 2020) research which proposed the use of quantile regression forests (QRF) as an anomaly detection system found in comparison to other methods, this approach was able to outperform others including similar approaches such as quantile regression (QR) and random forest (RF).

*M. Evangelou and N.M. Adams/Computers & Security 97 (2020) 101941*

**Table 1**

The errors of the methods are ranked from the smallest to the largest for each device of the ICL network. The ZIP approach did not fit for 13 devices. The sample mean of the ranks across all devices are presented.

Error	Benchmark	Poisson	ZIP	RT	RF	QR	QRF
MSE	5.7273	3.6000	4.2182	2.7818	4.6000	3.3455	3.7091
MedianSE	1.7636	4.9273	5.2545	4.3455	6.4727	2.6909	1.8909
MAE	3.0727	4.6727	5.2364	4.9818	6.3091	1.6000	2.1091
MedianAE	1.6545	4.5455	5.0000	5.3636	6.4727	2.5455	1.2727
MSAE	2.8909	4.9091	5.5091	4.3818	6.4000	2.2364	1.6727
MedianSAE	1.6545	4.6545	5.1273	5.2545	6.3455	2.4909	1.1636

**Table 2**

Computed errors across all 55 devices with the test set combined as a single set.

Error	Benchmark	Poisson	ZIP	RT	RF	QR	QRF
MSE	10099.6822	Inf	Inf	9435.4471	9097.8144	66677.5533	8332.3215
MedianSE	1.0000	1.0326	0.3127	1.5673	2.2107	1.0000	1.0000
MAE	1.3005	Inf	Inf	2.0924	5.6682	1.1889	1.2309
MedianAE	1.0000	1.2249	1.3912	1.4164	1.6419	1.0000	1.0000
MSAE	0.9338	0.9879	1.0782	0.9903	1.0425	0.9270	0.9002
MedianSAE	1.0000	1.1630	1.1713	1.1744	1.2260	1.0000	1.0000

Figure 3 - Error rate of multiple ML Methods (Lower is better)

Figure 3 displays results after testing and calculating the predictive accuracy of each algorithm. QRF was found to be the most effective in reducing the overall error rate in comparison to other algorithms across all devices.

**Table 4**  
The errors of the methods are ranked from the smallest to the largest for each device of the LANL network. The ZIP approach did not fit for 79 devices and QR did not fit for 74 devices. The sample mean of the ranks across all devices are presented.

Error	Benchmark	Pois	ZIP	RT	RF	QR	QRF
MSE	3.8429	4.9372	5.4607	2.8639	4.0576	3.7330	2.4241
MedianSE	3.0785	4.6911	5.3560	3.5812	5.5812	2.4869	1.8639
MAE	3.6126	5.1466	5.5288	3.4188	5.3560	2.4450	1.7539
MedianAE	2.6492	4.6335	5.1832	3.6492	5.3874	2.6178	1.8377
MSAE	3.9843	4.6492	5.3141	3.5183	5.1257	2.8272	1.8377
MedianSAE	3.4869	4.7644	5.1675	3.7696	4.9267	2.4241	1.5969

Figure 4 – Error rate reiterated

This is further reiterated in later research, Figure 4, where QRF outperformed other algorithms when tested on a larger number of devices.

## 2.2 Algorithms & Techniques

Machine learning implements various algorithms as part of different techniques like supervised and unsupervised learning. Each has a different use and will be more suited to the specific data you're working with which can have an overall impact on the outcome due to the suitability to the task at hand.

### 2.2.1 Unsupervised Machine Learning (UML)

(Saranya et al., 2020) Unsupervised machine learning uses clustering, association analysis or dimensionality reduction to predict the outcome based on unlabelled data. Since there is no labelled data, there is no training data provided in unsupervised machine learning.

This can be achieved through many different methods with the most common K-Means Clustering and Neural Networks.

#### 2.2.1.1 Neural Networks (NN)

Neural networks are the only machine learning algorithm to closely resemble the human brain. Similarly comprising of neurons to handle information. Using multiple layers of neurons allows data to be finely calculated before outputted (Kufel et al., 2023).



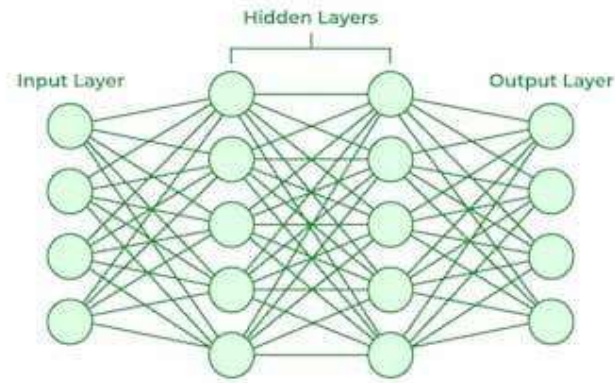


Figure 5 - Example neural network structure.

Shown in, Figure 5, is an example of how neural networks are structured with each node being interlinked and sharing information.

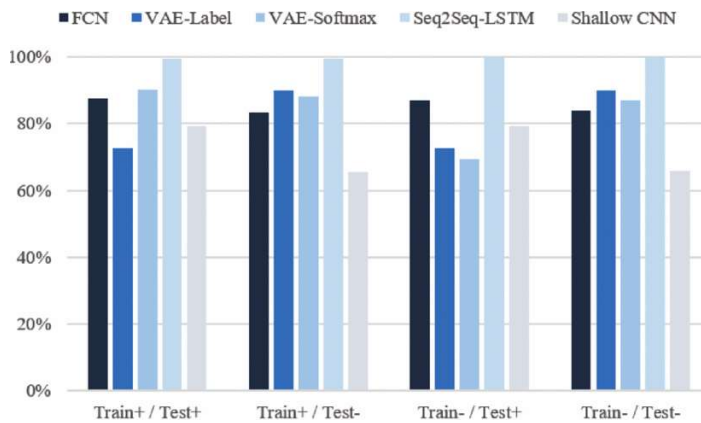


Figure 6 - Comparison of various Deep learning models (Kwon et al., 2018)

In a study on convolutional neural networks a common type of neural network to be used for machine learning applications, it was found that regardless of the node depth i.e. whether it is a shallow or deep CNN performance did not change significantly, Figure 6, yet did not work better than other algorithms such as LSTM(Kwon et al., 2018).

## 2.2.2 Supervised Machine Learning (SML)

Supervised machine learning works much like unsupervised machine learning in the sense that you have a machine learning model will produce an output except (Bhavitha et al., 2017) this uses pre labelled training data sets inclusive of the input data and expected output data.

### 2.2.2.1 K Nearest Neighbours (KNN)

K Nearest Neighbour uses a comparison algorithm based on current data the model has been trained on. This comparison algorithm plots data onto a graph including the new data. Using this comparison algorithm, it will review current data it knows and calculate the distance between points to find the near plot cluster, hence nearest neighbours.

Once a decision has been made on the nearest neighbouring cluster, this will then be classified accordingly to that cluster.



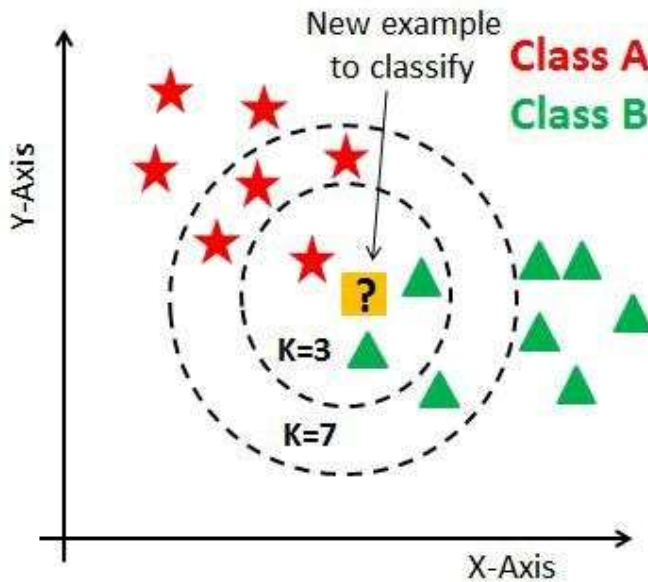


Figure 7 - Example KNN plot graph (Christopher, 2021)

(Christopher, 2021) for each round of machine learning that is run, a number must be assigned to K, this determines the number of points taken into consideration. Figure 7 displays an example of this, a new data point and two examples firstly where  $K=3$  and secondly  $K=7$ . In the first example, the new data point would be classified under class b as there are more data points from that class, similarly in the second, the new data point would be Class A as there are more data points in that cluster.

### 2.2.3 Ensemble Machine Learning (EML)

Ensemble algorithms are an amalgamation multiple algorithms from separate both supervised and unsupervised, or multiple instances of the same method. (Deitterich, 2002) explains how these take a different approach compared to other machine learning techniques and algorithms. Ensemble learning attempts to hypothesise more than a single solution in comparison to other non-ensemble machine learning algorithms.

#### 2.2.3.1 Quantile Regression Forests (QRF)

QRF at its base level is a mix of Quantile regression and random forests.

Quantile regression functions by splitting the data into quantiles, e.g. 0.25, 0.5 or 0.75 (Cleophas & Zwinderman, 2021); these can be looked at on a minority – majority scale where if a result has been attached to 0.75 it falls within a quantile along with 75% or the majority of other results. Then on the other hand, random forests aim to improve accuracy, prevent against overfitting and overall increase the robustness of the model by aggregating the outcomes from trees (Breiman, 2001).

QRF encompasses both of these techniques, using random forests to build decision trees out of multiple leaf nodes.

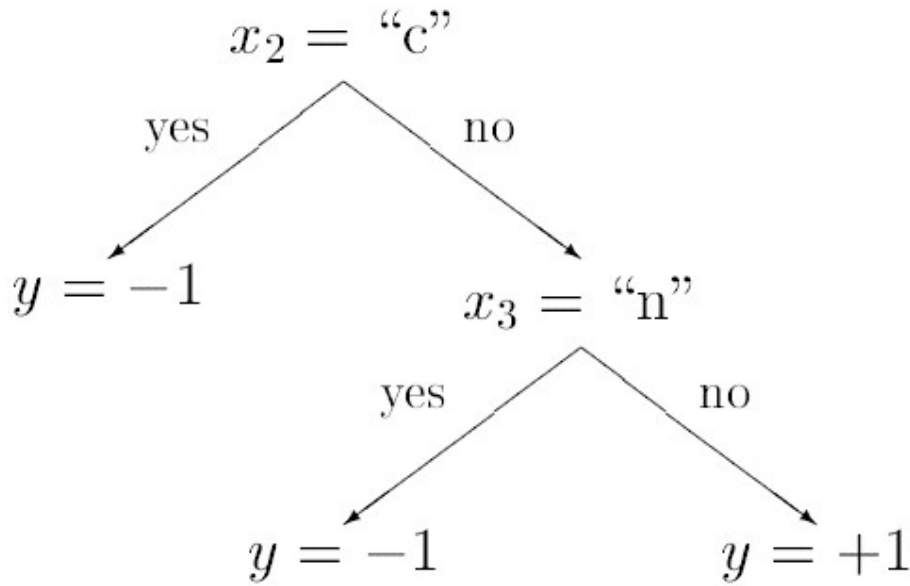


Figure 8 - Example Decision tree (Deitterich, 2002)

These decision trees then retain their predictions creating new predictions based on outcomes of previous leaf nodes.

### 2.3 Gaps in research

Threat detection with the use of federated learning over machine learning is still fairly new within the AI space. In comparison to the use of basic machine learning, the comparison having specific gaps in research with ML long having many algorithms implemented and researched versus FL which is still being developed.

There has been a specific amount of research detection using federated learning approaches, however the vast majority of this has been implemented in IoT environments for smart homes using algorithms such as neural-networks, KNN or custom with very little being introduced into a corporate or test environment for security.

None of the federated learning research articles observed used a dedicated, simulated vulnerable environment to create a realistic scenario, at most the dataset used will have been created either in a realistic or simulated environment but seldom was further testing regarding usability of the dataset within a realistic scenario performed.

### 2.4 Why Federated learning?

There are many reasons why federated learning has been chosen as the main construct over the likes of machine learning or deep learning for this project with one of the main reason its privacy preservation over other techniques. Although at its heart it uses machine learning models and methods to makes its predictions it does this in a way which improves security, network and hardware load and computation speed.

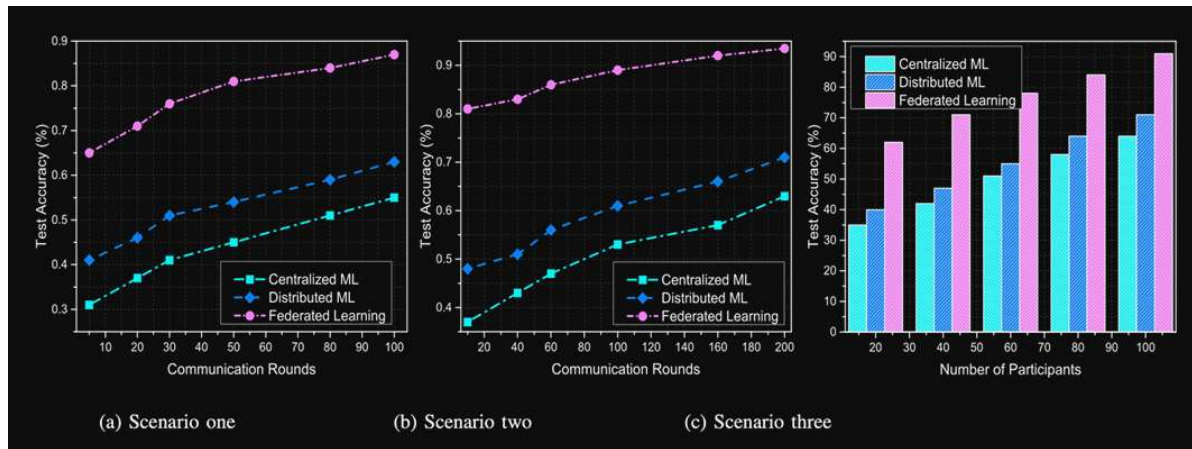


Figure 9 – Comparison of accuracy between machine learning and federated learning (Asad et al., 2021)

Explained further in (Asad et al., 2021) and in relation to Figure 9, the accuracy of federated learning is shown to be superior in comparison regardless of how many clients are with higher accuracy predictions taking place the more clients are involved.

### 3 Research Methodology & Design

#### 3.1 - Introduction

#### 3.2 - Research Methodology

This section will run through each stage of the project, setting out the basis for how each section will be performed and basic expectations.

##### 3.2.1 – Data procurement and importing

###### 3.2.1.1 – Dataset requirements

When considering the model for federated learning, the base of the model is the data it is trained tested and validated against. To ensure the best model for predictions, dataset(s) used have requirements, looking specifically at the relevancy of features and labels, accuracy of data, source of data, balance of labels and if training on multiple datasets the size of datasets.

###### 3.2.1.1.1 – Dataset Relevance

Dataset relevancy plays a vital role in model development and directly affects the overall accuracy according to the data it is being tested on. Relevant datasets are those which pertain to the proposed use case, i.e. a model for predicting BTC stock market rises/crashes would be trained using prior BTC stock market data not ETH data as this would reduce the accuracy of the model due to it lacking relevancy to trends relating to that specific market. Relevancy also concerns how realistic data being used is as this is what it will be analysing once trained.

###### 3.2.1.1.2 – Accuracy and Completeness

Accuracy of the dataset is also vital for creating a model that provides precise predictions. The lack of accuracy within datasets used to train the model such as a label identifying malicious behaviour as benign would degrade and have a significant impact on overall model accuracy.

###### 3.2.1.1.3 – Class Distribution

Within the overall training data, having a majority of data classified as malicious will severely impact the overall accuracy of the model and effectiveness distinguishing between malicious and benign

behaviour. With the correct distribution of true positive and true negative, the model should sufficiently be able to predict these correctly and unknown.

#### 3.2.1.1.4 – Size

Overall dataset size can also create issues when training the federated learning model, potentially having a significant effect on model performance. Aggregating multiple datasets to create a superset can introduce resource issues with the larger the dataset the more resources needed to process the data, the larger dataset can also extend the time to train the model. On the other hand, datasets on the smaller side pose their own issues to the model, most importantly the impact on prediction efficiency. Smaller datasets could mean the model does not have an expansive enough knowledge to be able to sufficiently class new events. Over time this reduces the efficacy of the model, reducing accuracy and creating the possibility that malicious events could be missed.

### 3.2.2 – Data preprocessing

Pre-processing data is crucial to both ML and FL projects. Considerations made during this were looked over three individual steps, data cleaning, encoding then normalisation. The use of specific encoding and normalisation steps will be justified within this section.

#### 3.2.2.1 Data Cleaning

As the dataset being used is one created by the University of New South Wales, UNSW-NB15 (Moustafa et al., 2017, 2019; Moustafa & Slay, 2015, 2016; Sarhan et al., 2021), using the IXIA PerfectStorm tool and not one created by myself it's impossible to know all values are as expected especially when attack data has been simulated. For this reason, it's important the data was checked for both null values and data that doesn't provide any impactful prediction value using the `isnull()` and `isunique()` commands. The implementation of data cleaning helps to ensure that the data is prepared, in the correct format and can be used for accurate predictions.

#### 3.2.2.2 Encoding

With future improvement of the project in mind, three encoding methods were considered for this project, Onehot, label and binary but after careful consideration Onehot was chosen based on its compatibility. Onehot encoding is suitable for algorithms which require numerical data such as logistic regression, preserves the original information and is fully compatible with federated learning where implementations of clients with separate datasets containing various different fields is handled well by this particular encoder.

(*Sklearn.Preprocessing.OneHotEncoder — Scikit-Learn 1.4.2 Documentation*, n.d.)

#### 3.2.2.3 Normalisation

Normalisation was also an important consideration with the mass amount of data contained within the dataset, this is done to provide higher accuracy for predictions with two scalers were considered in this instance, minmax and robust. The robust scalar was chosen over the minmax scalar for a couple of reasons, firstly the robust scalar is not as sensitive to outliers which could skew data during the normalisation process, secondly unlike other methods data is not centred around the mean value of data which can lead to variance in results or inaccurate in cases where numerical data is meaningful, for instance port numbers which don't need a mean average.

(*Sklearn.Preprocessing.RobustScaler — Scikit-Learn 1.4.2 Documentation*, n.d.)

### 3.2.3 Results

Once all the results are collected, two things will happen to them. Firstly, a confusion matrix graph will be produced, this will be used to gather statistics on true positive, false positive, true negative and false negative results. Sci-kit learn includes a module for this confusion matrix which will be used to document the efficacy of federated learning for threat detection. Secondly, results should be passed through to the SIEM infrastructure for alerting.

(Sklearn.Metrics.ConfusionMatrixDisplay — Scikit-Learn 1.4.2 Documentation, n.d.)

### 3.3 Virtual Machines

Building the virtual SOC environment for testing, it's important to have various operating systems, versions and vulnerability levels to create a variety of potentials in the threat landscape. It's also important to consider what is most used within organisations globally with both Linux and Windows systems dominating the corporate market(*Linux, the Operating System of Choice*, 2019), these would be the best system to perform tests on, also, MacOS is not an openly available operating system so is not possible to test without the purchase of an Apple computer.

The following virtual machines will be provisioned for testing:

Machine (OS)	Usage
CentOS7 (Free)	Standalone various services such as SSH, Samba enabled
Ubuntu 14.06 (Free)	Vulnerable machine, No firewall, All ports open
Windows Server 19 Datacentre (Free licence from university)	Hosting various services such as active directory

(CentOS7, n.d.; Ubuntu Releases, n.d.)

### 3.4 – Ethical Considerations

One major consideration of this project is the ethical consideration surrounding data collection and processing data which has been collected and that which is in the dataset(s). Any dataset used within the project is opensource, open to the general public and can be freely accessed via the internet; these are also cited and credited appropriately throughout. Any data collected is done so within a controlled environment, any users which attack the system are attacking at their free will. No personal information is collected or processed as part of the project. All data will be removed, permanently deleted from all sources at the end of the project.

### 3.5 – Search Methodology

When obtaining dataset(s) for use in model training and testing, it was important to acquire a range of data which would serve and fit model requirements. Past research papers played a key role in this as researchers detailed the features of datasets used within their papers. Kaggle and GitHub also proved to be rich sources, Kaggle where users upload datasets with machine learning projects, they used them on and a specific GitHub page (reference) which lists many cybersecurity related datasets for multiple use cases.

When searching for datasets terms such as “intrusion detection dataset”, “NIDS dataset”, “HIDS dataset” and “cyber security anomaly detection dataset” were used. This is to rule out any non-cyber related datasets such as MNIST.

### 3.6 – Model Algorithm Requirements

When considering the model algorithm, it was important to consider both the task at hand and how appropriate the selected algorithm is for the general use case, i.e. anomaly detection. Algorithms unoptimized for the use case could result in potentially skewed or inaccurate results. The following section will cover differences between alternative methods which can be used to derive a suitable algorithm based on previous research within the literature review.

### 3.7 – Supervised, Unsupervised or Ensemble Learning

Within machine learning there are three main types of method and as mentioned previously when talking about algorithms, it's important to cater the method to the issue and this can be dependent on various factors, for example the data involved. Will there be a training dataset, if so, the method needed will likely be more suited to supervised or ensemble techniques. These will utilise labelled data, to train the model and henceforth predict new unknown data. This type of learning is commonly implemented for classification and regression algorithms where an outcome (label) is predicted based on inputs (features).

Unsupervised learning on the other hand only deals with unlabelled data and from this will attempt to learn the data itself, deriving any relations to form relevant and accurate predictions. This may be easier to implement but at the sacrifice of accuracy. With the project requiring accuracy to predict malicious intent, unsupervised machine learning wouldn't be a great fit but supervised or ensemble machine learning would be with an ensemble method such as quantile regression forests being a generalisation of random forests and quantile regression.

Based on firstly the dataset for the project being labelled and secondly the need to predict the value, the algorithm chosen to work with this data would need to be suitable. Supervised regression algorithms such as logistic regression and quantile regression forests provide accuracy with fast paced predictions or for a slight trade off in speed Neural Network Regression improves accuracy.

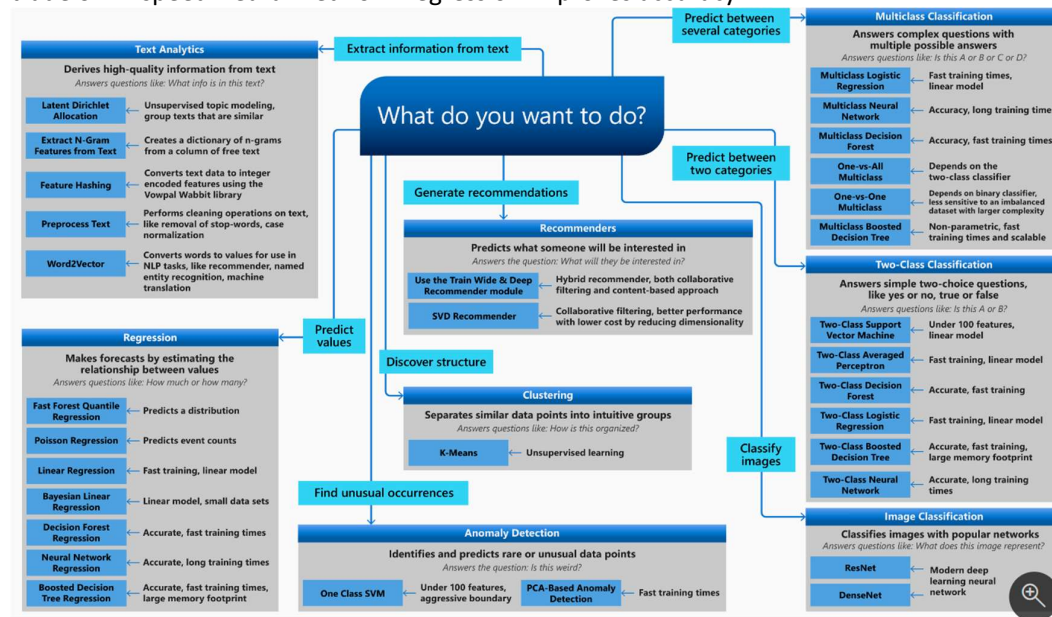


Figure 10 – Azure ML Algorithm Sheet (Gayhardt, 2023)

### 3.8 – Virtual Environment

The following sets out the hardware and software specifications used for the project outlining all components inside the machines used for the FL tasks along with libraries and programming languages installed for creating the model.

### 3.9 – Hardware Specifications

Hardware is an important consideration for a machine learning project due to the type of task being carried out. Especially with federated learning, running multiple clients and a server, creates extra stress on the system in comparison to traditional machine learning. Using cloud services would have introduced limitations to the project such as not having the ability to work with local files. Two systems I already had were good enough for federated learning removing any potential constraints of cloud services from the project, however the main system used for the project did fail completely at the main stage of the project.

Custom Laptop from Lenovo

CPU: Intel Core 12-Core i7-9750H 2.6GHz turbo 4.5GHz.

SSD: 2x 1TB M.2 NVME (Raid 0).

RAM: 32GB 3200Mhz DDR4.

GPU: Nvidia Quadro T2000.

Custom Desktop:

CPU: AMD Ryzen 5 3600 Overclocked (NZXT Watercooled) (Failed)

M.2 SSD: Samsung 970 Evo Plus 1TB (Main Drive) (Previously Fanxiang 1TB M.2 Failed and replaced)

M.2 Heatsink: Sabrent Rocket 2280 (White)

SSD(RAID0): Fanxiang S101 1TBx2 (Games)

SSD: Sandisk 250GB (Testing Environment)

HDD: Western Digital Blue 5400rpm 1TB (Video Editing)

RAM: Corsair Vengeance LPX 48GB 3200mhz

GPU: XFX 5600XT Thicc 3 Ultra

### 3.10 – Software Specification

Firstly, when beginning the project Jupyter notebook was used as I was familiar with this, and python 3.12. However, the switch to Visual Studio Code was quickly made as notebooks did not work as intended with the code created, for the way I wished to work with it and python 3.12 was not compatible with libraries being used. 3.12 was traded for 3.11.4, which provided the ability to use important data manipulation libraries pandas and NumPy, graphing libraries confusion matrix and matplotlib and the main libraries being used throughout the project for machine learning Scikit-Learn (sklearn) (Buitinck et al., 2013; Pedregosa FABIANPEDREGOSA et al., 2011) and federated learning flwr(Beutel et al., 2020).

### 3.11 – Conclusion

This chapter covers the considerations during decision process of choosing model algorithms for the project beginning with the extracting dataset requirements needed to predict required outcomes along with desired algorithms capable of carrying out such tasks. Creation of supervised regression models were chosen due to the overall effectiveness and match to the task. Hardware and software requirements are also clearly set out here.



## 4 – Results & Discussion

Throughout this section the federated learning code will be covered, explaining crucial parts of the process such as pre-processing and encoding. Progression will be reviewed throughout the project and a conclusion will be drawn at the finalisation of the project looking at all work that has been carried out followed by a review of future work and a personal review reflecting on any setbacks with the negative impact this had on the project such as any limiting factors.

### 4.1 Windows server setup

Initial setup of the windows server included renaming the server to a suitable name “WindowsDC” and setting a static IP address. After this point active directory and DNS roles were installed on the server along with creating a new forest “adamfl.com” so the active directory domain can be discovered by other computers on the network.

### 4.2 Vulnerable Machine Setup

This section covers the setup of the SOC environment consisting of 2 linux machines and two windows machines. One of those windows machines used as an active directory server. The general setup will not be covered; however, all machines will be provisioned with 4gb of ram, 1 CPU 2 cores and a bridged network connection in range 192.168.0.13x

#### 4.2.1 Ubuntu VM Creation

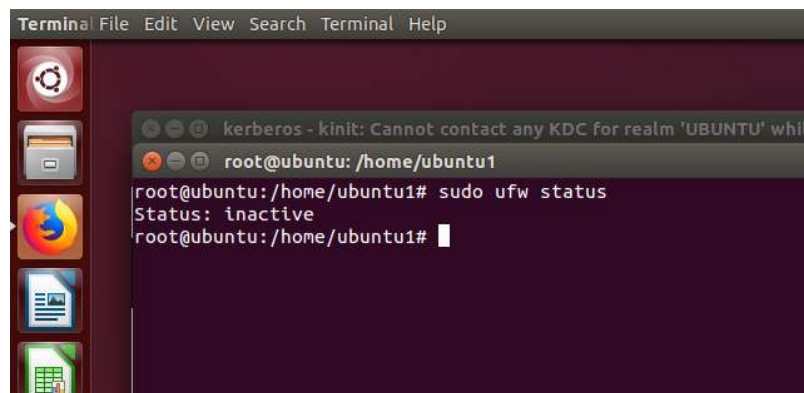


Figure 11 - Ubuntu version 14 inactive firewall - open ports.

After setting up the first ubuntu machine running ubuntu version 14, I then checked the status of the firewall using `sudo ufw status`, Figure 11. This is naturally inactive showing the firewall is inactive with all ports open which is exactly what is needed for the first machine.

#### 4.2.2 CentOS VM Creation

For the second machine the setup is a bit more complex. This requires a machine connected to active directory and for this I chose to use CentOS as I have previous networking experience with this specific OS.

```
[cent_root@cent_root ~]$ sudo yum -y install sssd realmd oddjob oddjob-mkhomedir adcli samba-common samba-common-tools krb5-workstation o  
penldap-clients policycoreutils-python
```

Figure 12 – installing active directory dependencies.

Shown in, Figure 12, I installed the dependencies first, this consisted of x.



```
[cent_root@cent_root ~]$ realm discover 192.168.0.135
```

Figure 13 – Discovering the AD server.

```
[cent_root@cent_root ~]$ realm join -v --user=WindowsDC 192.168.0.135
```

Figure 14 – Connecting to AD server logging in with the DA user.

After this I used realm discover [Server IP] to find the active directory server I had previously setup, Figure 13, then used realm join -v -U WindowsDC [Server IP] to join the active directory, Figure 14.

```
PS C:\Users\WindowsDC> Get-ADComputer -filter "name -like 'cent_root'"

DistinguishedName : CN=CENT_ROOT,CN=Computers,DC=adamf1,DC=com
DNSHostName       : cent_root
Enabled           : True
Name              : CENT_ROOT
ObjectClass       : computer
ObjectGUID        : 5b5ca2d7-4bbb-4a55-b0d0-839690a086ce
SamAccountName    : CENT_ROOT$
SID               : S-1-5-21-947489694-3196831738-138024384-1105
UserPrincipalName :
```

Figure 15 – CentOS VM shown in active directory (windows)

This is confirmed by looking at active directory where the CentOS machine can be seen Figure 15.

### 4.3 Kali Setup

For testing I also setup a kali linux machine, unlike the other linux machines, there was no extra setup for this, but this was connected to the same IP range.

### 4.4 SIEM Implementation

Unfortunately, SIEM software Siembool was unable to be implemented due to issues during setup on the ubuntu machine. The Siembool installation first encountered issues due to file configurations and dependency issues, however, after the correct versions of libraries were downloaded and configuration files altered to suit system specifications a Kubernetes cluster was created. After this step the main issues were setting up storm topologies and setting up zookeeper nodes.

### 4.5 Quantile regression implementation

Whilst attempting to implement the Quantile Regression Forests (QRF) model into federated learning, which was the original plan, this is when I found out that I would be unable to do so due to QRF not being directly compatible with the Scikit-Learn framework causing complications within the code.

This was not initially clear during the beginning of my research as scikit-learn currently implements quantile regression, however, when attempting to implement the QRF model it was made clear there was much more time needing to be allocated for troubleshooting and fixing code than originally estimated.

### 4.6 Logistic Regression Model

After being unable to implement QRF the choice was made to move away from that approach and instead use a logistic regression approach, which at this point in time after having system issues would also be more

fitting with new hardware limitations and time restrictions of the overall project from this point going forward.

Logistic regression should still provide a key insight into how federated learning can impact the threat detection process without sacrificing much of the original intended prediction accuracy in comparison to the original QRF method.

## 4.7 Initial approach

### 4.8 – Importing Data

```
tabnine: test | fix | explain | document | ask
def load_Data() -> Dataset:

    #Loading Datasets
    trainDF = pd.read_csv(r"C:\Users\adamc\Work\HomeLab\ML Dissertation\Datasets\UNSW-NB15\UNSW_NB15_training-set.csv")
    testDF = pd.read_csv(r"C:\Users\adamc\Work\HomeLab\ML Dissertation\Datasets\UNSW-NB15\UNSW_NB15_testing-set.csv")
```

Figure 16 - Loading training and testing datasets.

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports	attack_cat	label
600	601	0.019387	udp	dns	CON	2	2	130	162	154.742874	...	1	1	0	0	0	1	1	0	Normal	0
601	602	0.001038	udp	dns	CON	2	2	132	164	2890.173445	...	1	3	0	0	0	17	2	0	Normal	0
602	603	28.046524	tcp	-	FIN	16	18	1272	2572	1.176616	...	1	1	0	0	0	12	13	0	Normal	0
603	604	0.715747	tcp	-	FIN	16	18	1540	1644	46.105677	...	1	5	0	0	0	10	17	0	Normal	0
604	605	1.335132	tcp	-	FIN	30	42	2302	34406	53.178262	...	1	5	0	0	0	10	17	0	Normal	0
605	606	20.558563	tcp	-	FIN	16	18	1272	2572	1.605171	...	1	1	0	0	0	3	3	0	Normal	0
606	607	0.911283	tcp	-	FIN	16	18	1540	1644	36.212680	...	1	10	0	0	0	14	16	0	Normal	0
607	608	1.250357	tcp	-	FIN	16	18	1540	1644	26.392462	...	1	5	0	0	0	14	10	0	Normal	0
608	609	20.166878	tcp	-	FIN	28	28	4656	2976	2.727244	...	1	2	0	0	0	17	7	0	Normal	0
609	610	0.981115	tcp	-	FIN	16	18	1540	1644	33.635201	...	1	5	0	0	0	15	17	0	Normal	0

0 rows x 45 columns

Figure 17 - Sample records from trainDF.

The beginning of the project started with the import of the chosen UNSW-NB15 dataset. The dataset chosen comes with two files training.csv and testing.csv which were imported using the pandas read.csv function, assigning the data to the relevant dataframe variable shown in Figure 16 with an example of the raw data this presents in Figure 17.

#### 4.8.1 – Dataset limitations

With the UNSW-NB15 dataset not being collected in a personal real-life scenario or as part of a company it is natural that the dataset is limited in terms of its exposure to both network and machine security risks. A real-life scenario would be conducted using prior and continuous data collected either in a corporate or personal environment dependent on where the model is being used. This data would include personally identifiable information (PII) such as real IP addresses, machine addresses, name, email addresses, user information network information, application information etc.

This would create a more bespoke model specifically designed for that environment and would also include multiple dataset sources for more information, further improving the overall performance of the model these datasets would be used to train. Unfortunately, due to the time restrictions and other factors, further explained in Setbacks & Challenges, only the UNSW-NB15 dataset was used for both training and testing as it provided separate data for both datasets but was the best, up to date dataset that could be found at the time of research which encompassed multiple attack values, was publicly available and included both network and machine attack data.

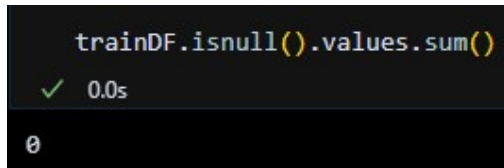
## 4.9 Pre-Processing

Pre-processing was carried out in multiple steps which included following basic steps of data pre-processing running through data cleaning, encoding and normalisation to ensure the dataset is ready to work with the chosen model.

### 4.9.1 Cleaning

Cleaning of data is carried out on datasets for many specific reasons but mainly to remove any missing (null) values, removing duplicates and removing any random outliers.

### 4.9.2 Entries with null values



```
trainDF.isnull().values.sum()
✓ 0.0s
0
```

Figure 18 - Checking for null values using isnull()

Removing null values improves the accuracy of the overall model by keeping only relevant and useful training data since null values do not provide any information for the model. For this reason, this check was only performed on training dataframe with the result returning 0, Figure 18, showing there are no null values in the training dataframe.

### 4.9.3 Entries with no unique values

The next part of the cleaning process required me to check that there were no unique values within the dataframe. This is to remove any features which would not provide any impact to the training data. This was carried out using nunique() again focusing specifically on trainDF to produce the following features with their corresponding unique records Table 1.

Table 1 - Number of unique records per dataset feature

Feature	No. Unique
id	175341
dur	74039
proto	133
service	13
state	9
spkts	480
dpkts	443
sbytes	7214
dbytes	6660
rate	76991
sttl	11
dttl	6
sload	80885
dload	77474
sloss	409
dloss	370
sinpkt	76161

dinpkt	74245
sjit	77532
djit	76831
swin	13
stcpb	75265
dtcpb	75089
dwin	7
tcprrt	43319
synack	40142
ackdat	37708
smean	1357
dmean	1328
trans_depth	11
response_body_len	2386
ct_srv_src	52
ct_state_ttl	5
ct_dst_ltm	50
ct_src_dport_ltm	47
ct_dst_sport_ltm	32
ct_dst_src_ltm	54
is_ftp_login	4
ct_ftp_cmd	4
ct_flw_http_mthd	11
ct_src_ltm	50
ct_srv_dst	52
is_sm_ips_ports	2
attack_cat	10
label	2

This shows there are no unique values in any of the features across the dataset which is.

```
trainDF['is_sm_ips_ports'].unique()
✓ 0.0s
array([0, 1], dtype=int64)
```

With a further look into 'is\_sm\_ips\_ports' by using the .unique() function, it's shown to only have two records. These are unusual ports and generally reserved ports so if an attacker were to access them, this would be an extremely suspicious event and as shown previously by (Luchs & Doerr, 2019)

#### 4.9.4 Redundant Value Check

At this point we're ready to remove any redundant values from both datasets, this is inclusive of potential null values that have been missed from the previous check, removing any duplicates and removing specific columns which are not needed.

#### 4.9.5 Removal of null and duplicates

```
#Dropping null and duplicate values
trainDF = trainDF.dropna()
trainDF = trainDF.drop_duplicates()
```

Figure 19 - Removing null and duplicate values from training set.

To remove any potential duplicate values that were not detected during the checking phase, although there shouldn't be any for peace of mind. Both `dropna()` and `drop_duplicates()`, shown below in Figure 19, were used specifically on the training dataset which will remove those records from the dataset to produce a more accurate training dataframe.

```
#Dropping redundant columns
trainDF.drop(labels="id", axis=1, inplace=True)
trainDF.drop(labels="label", axis=1, inplace=True)

testDF.drop(labels="id", axis=1, inplace=True)
testDF.drop(labels="label", axis=1, inplace=True)
```

Figure 20 - Removing redundant columns.

After removing the duplicated I then focused on removing labels, shown in Figure 19 - Removing null and duplicate values from training set., which would not provide any valuable contributions to the dataframe itself. The majority of the UNSW\_NB15 dataset contains relevant information that either adds to detecting network or machine attack data so the decision was made to only remove ID and label. Both TrainDF and TestDF have their own ID when it is created and the label only supplies two outcomes, 0 (non-malicious) or 1 (malicious), whereas `attack_cat` provided ten outcomes such as if activity was Normal, Denial of service or Reconnaissance.

#### 4.9.6 One hot encoding

##### 4.9.7 What is it?

Simply, one hot encoding is used to change records in the dataframe to readable data for the machine learning algorithm. An example of this would be `attack_cat` from the dataset, as this is a feature that contains words, the logistic regression algorithm cannot use it since it only accepts numerical data.

#### 4.9.8 Implementation

```
# Apply one-hot encoding
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
enc.fit(trainDF[['proto', 'service', 'state', 'attack_cat']]) # Fit encoder on training data
```

Figure 21 – Fitting encoding to training dataset.

There are multiple stages to the One hot encoding, firstly shown in Figure 21 – Fitting encoding to training dataset. is creating the encoder. To do this a variable is created in this instance "enc" which is assigned the one hot encoder and then uses `trainDF` to change each feature to a numerical value.

```
train_encoded = enc.transform(trainDF[['proto','service','state','attack_cat']])
test_encoded = enc.transform(testDF[['proto','service','state','attack_cat']])
```

Figure 22 - Encoding columns of both datasets.

After fitting the encoder to the training data, this is then used to transform the unreadable features to readable ones for the model shown in Figure 22.

```
# Replace 'proto','service','state','attack_cat' column with encoded data
trainDF = pd.concat([trainDF.drop(['proto','service','state','attack_cat'], axis=1), pd.DataFrame(train_encoded, columns=enc.get_feature_names_out(['proto','service','state','attack_cat'])), axis=1])
testDF = pd.concat([testDF.drop(['proto','service','state','attack_cat'], axis=1), pd.DataFrame(test_encoded, columns=enc.get_feature_names_out(['proto','service','state','attack_cat'])), axis=1])
```

Figure 23 - Replacing existing columns with new OHE columns.

Then finally to complete the encoding process, the existing proto, service, state and attack\_cat features within the dataset were replaced with the encoded versions by concatenating using pandas shown above in Figure 23 with example differences seen in Figure 24 to Figure 25. Rather than there now being 43 features in the dataframe, there are now 204.

attack_cat
Normal
Normal
Normal
Normal
Normal
...
Generic
Shellcode
Generic
Generic
Generic

Figure 24 - Original Attack Category

Below in Figure 25 is the onehotencoded attack\_cat. It displays each individual category as a feature, with a numerical value assigned 0-1 to show if this applies to specific feature applies to that record in the dataframe or not.

attack_cat_Analysis	attack_cat_Backdoor	attack_cat_DoS	attack_cat_Exploits	attack_cat_Fuzzers	attack_cat_Generic	attack_cat_Normal	attack_cat_Reconnaissance	attack_cat_Shellcode	attack_cat_Worms
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0

Figure 25 - OneHotEncoded Attack Category

#### 4.9.9 Sorting labels and features

```
#Create classes variable listing all outcomes of dataset
class_name = ['attack_cat_Analysis',
'attack_cat_Backdoor',
'attack_cat_DoS',
'attack_cat_Exploits',
'attack_cat_Fuzzers',
'attack_cat_Generic',
'attack_cat_Normal',
'attack_cat_Reconnaissance',
'attack_cat_Shellcode',
'attack_cat_Worms']

# Select everything other than classes
x_train = trainDF.drop(columns=class_name)
x_test = testDF.drop(columns=class_name)
# Select only classes
y_test = testDF[class_name]
y_train = trainDF[class_name]
```

Figure 26 - Assigning new attack\_cat labels to Y variables.



	attack_cat_Analysis	attack_cat_Backdoor	attack_cat_DoS	attack_cat_Exploits	attack_cat_Fi
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
82327	0.0	0.0	0.0	0.0	0.0
82328	0.0	0.0	0.0	0.0	0.0
82329	0.0	0.0	0.0	0.0	0.0
82330	0.0	0.0	0.0	0.0	0.0
82331	0.0	0.0	0.0	0.0	0.0

82332 rows × 10 columns

Figure 27 – Y variable example showing only labels assigned.

Now that the new labels are created, the next step is to separate these between X and Y variables with X being used for features and Y used for the label. This is done by first creating an array of all 'attack\_cat\_\*' and assigning that to a class\_name variable. Class\_name is then used to create x\_train/x\_test by dropping those specific columns from the dataframe and y\_train/y\_test by assigning class\_name to it directly which drops all other columns as shown in Figure 26 with an example in Figure 27.

#### 4.9.10 Normalisation

With X and Y variables now created and relevant data now assigned to them the final step of preprocessing can be carried out. Normalisation is performed on y\_train and y\_test using robust scaling a technique that takes the dataframe specified, converting data to the interquartile range (IQR) the first quantile and third quantile and removing the median. Although this is done currently on data where normalisation is not needed due the range being 0-1, if a new dataset was implemented this feature would already be in place and the new dataset would work as expected.

### 4.10 Revised Code

This can be found in Appendix C: Code

After creating and testing the previous code some amendments needed to be made after updated to the flower federated learning module which changed many of the ways the client and server interacted with the utils file, there was less dependency on this and deprecated functions.

The revised code at this stage is the same as the previous minus the manual onehotencoding, this is carried out via pandas using pd.get\_dummies(). This revised code removed inhomogeneous data errors previously being seen and provided code that was not only updated, but easier to work with and worked flawlessly every time.



## 4.11 Results

## 4.12 Training

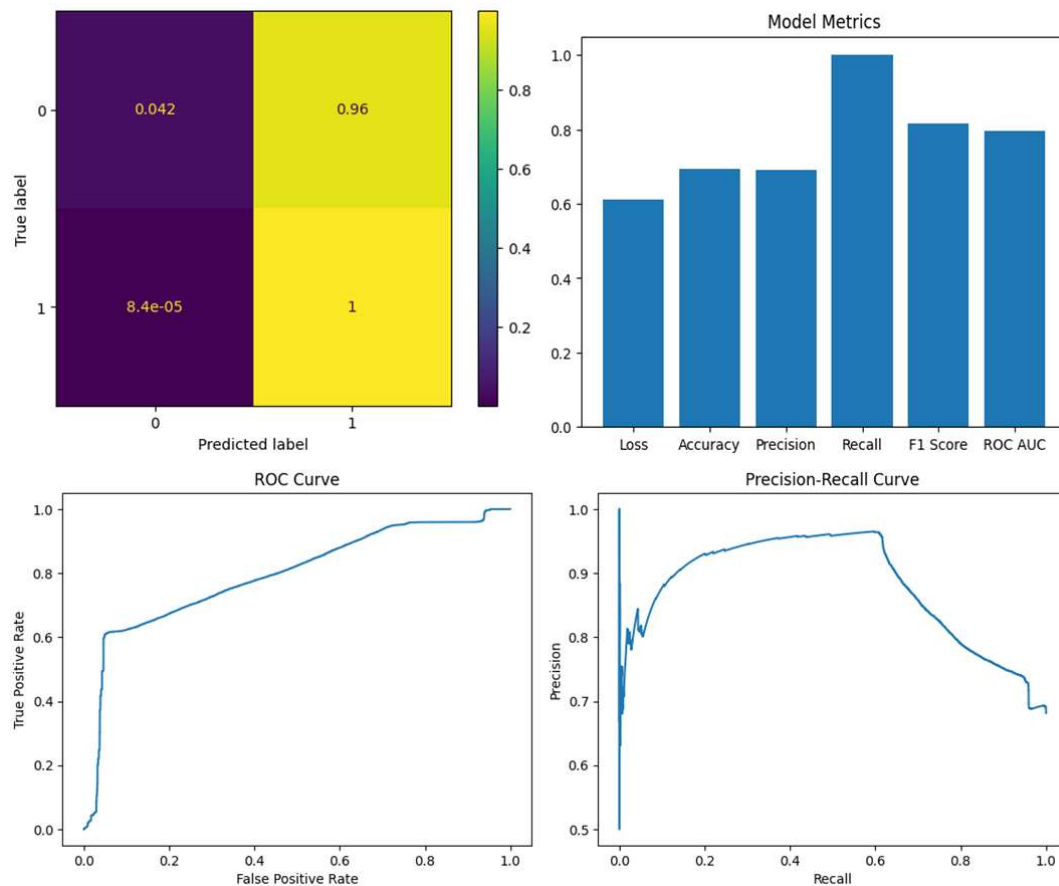


Figure 28 – Training metrics (CM, MM, ROC, PR)

Beginning with the metrics from the training phase and creating the model, shown in Figure 28 which consists of Loss, Accuracy, Precision, Recall, F1 Score, and ROC AUC metrics.

These are the results from completing 4 sets of federated learning. Each set consisted of 10 clients and 30 rounds. Data was partitioned into 50 partitions with 10 partitions being looked at per set i.e. first set 0-10 second set 10-20. A final 6th set was then carried out with no partitioning, 10 clients and 50 rounds, this is much like a five-fold cross-validation where five sets of partitioned training data is used to train the model before running a final pass on the dataset.

The confusion matrix here displays a high rate of detection for true negatives shown by the bottom right square (1) and a low false positive rate shown by the bottom left square (0.000084). Both of these metrics suggest that the model overall should be effective at identifying outliers to the model since it understands what a true negative should look like. However, there is also a worry here that the high score here could be as a result of overfitting the model.

Moving onto the model metrics bar graph, this display results for everything in its entirety and here we are expecting a high bar in everything other than loss where lower is better. At the end of training the model had an average loss rate of 0.61 and an accuracy of 0.69 on training set. Although this accuracy is good, the loss rate is worrying as this leans more into the model being over fitted.

F1 score, and recall does well here also, however recall and precision go hand in hand to predict true positives and in the PR graph, precision can be seen lowering after reaching just over 90%. In a similar situation is ROC where although indicating effective performance, the false positive rate steadily climbs.

#### 4.13 Testing

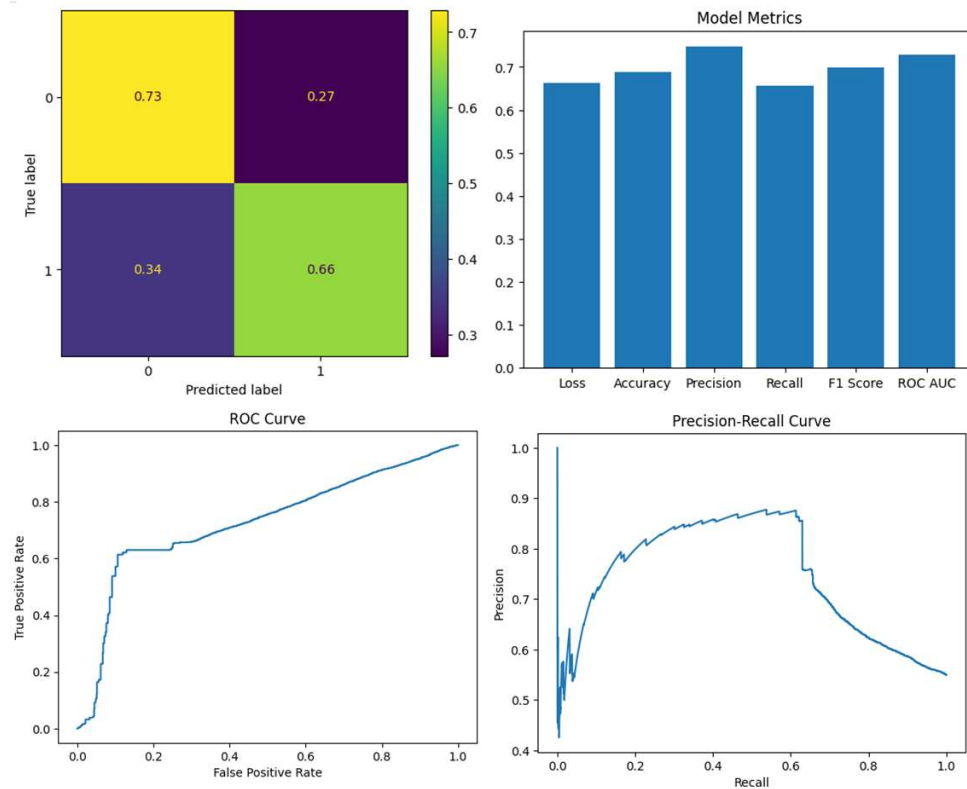


Figure 29 - Testing metrics (CM, MM, ROC, PR)

When looking at the graphs, Figure 29, after testing there is a stark difference.

The new confusion matrix indicated that there is an issue with the model, in its current state, accurately predicting the if this is malicious activity or not. Although 66% of true positives and 73% of true negatives were correctly predicted, both false positives and false negatives were both higher than expected.

This leads onto the model metrics where all metrics have near enough evened out which wouldn't be too concerning if loss was nearly as high as all other metrics gathered. This is more noticeable in the precision-recall graph where previously precision was over 90% yet now it doesn't reach that. In contrast to these negatives however, ROC displays an increase coinciding with the CM's increase in identifying true positive.

#### 4.14 Conclusion, Reflection & Future Work

Removing the negative aspects, the project did overall go well, working federated learning code was produced using the UNSW\_NB15 dataset. I believe the outcome of this project does show the potential for the use of federated learning within a security environment for analysis of both network and machine threats using distributed machine learning algorithms for the categorisation of malicious activity.

Overall, the model performance fared well, 0.6 to 0.7 accuracy is deemed an okay standard (Allwright, 2022) and it is clear from the results produced there is still room for improvement. The testing phase displayed good promise the model would perform well once trained. It seemed to accurately identify negative results with its higher true negative and lower false positive scores.

Moving onto testing, this displayed less than expected performance as shown by both the model metrics and precision-recall graphing. Accuracy stayed consistent throughout both phases but loss increased from 61% to 65%, with recall decreasing drastically. This overall means one of three things needs to happen going forward, either: The code needs to be optimised, implement a different ML algorithm or implement a different FL/ML framework. Going forward I think these points are the most important ones to consider when looking into future improvements.

#### 4.15 Setbacks & Challenges

##### 4.15.1 System Issues

Beginning with looking further into some of the challenges faced throughout the project, I encountered several system related issues.

Beginning in November, challenges arose when my desktop, the main computer being used for this project, began to shutdown unexpectedly. Unfortunately, this led to the M.2 SSD failing in December causing further delays to the project. The M.2 was replaced at the start of January 2024, fixing the previous issues of the system but not the loss of virtual machines, dataset and most recent changes to code and dissertation.

However, again in February the desktop failed with CPU issues. At this time it was not possible to fix these issues and the decision was made to move completely over to my secondary pc and an attempt to recover data from the primary was made using an external SSD enclosure, however, this was unsuccessful. This time proving a significant impact to project development and resulting again in loss of datasets previously gathered as this data was too large to backup to cloud solutions, virtual machines and the most recent changes plus any research downloaded; this system to date (01/05/2024) has never recovered.

After these two issues I was then left with my laptop with older hardware than the main computer. I also faced issues with this as my laptop specification was not entirely up to the task of the original project. Laptop hardware as standard is worse than desktop hardware due to many different issues, thermal issues being one of them which mainly limits CPU, the main component being used in this project. Adding onto this, scikit-learn does not support GPU acceleration unlike TensorFlow. This originally wasn't an issue with the main system, as the GPU was an AMD GPU and wasn't supported for GPU acceleration, but the CPU was more powerful than using my laptop so a trade-off was made, however, this proved to be a limiting factor in the long run.

##### 4.15.2 Networking Constraints & Documentation Challenges

Network constraints also presented a significant impact to the project. Due to the Carrier-Grade Network Address Translation (CGNAT) my ISP utilises this meant I could not use the public IP address of my network. To gain access to this would be difficult and without special consideration from the ISP providing me with my own static or dynamic IP range, this would be n/e on impossible. This network infrastructure drastically changed the direction of the project, again producing further delays as a new solution would need to be implemented. In January, the decision was made for the project to be localised to my main system.

Additionally, poor documentation and code maintenance throughout the duration of the project exhausted a considerable amount of already precious time. When configuring SIEM software it was apparent it was not documented correctly and considerably out of date with most sections not listing dependencies, code used for initial setup not correct and requiring mass amounts of editing to get it at least partially working in both windows and Linux environments. Once installation code was working, Installing helm charts and configuring

Kubernetes became the main problem, again another project changing issue as the combination between lack of documentation and time constraints meant configuring the SIEM would not be feasible at this point in the project.

#### 4.15.3 Health Issues

Throughout the project I dealt with new health issues, taking time out extensively due to misdiagnosis and stress which made my condition worse, working from home due to social anxiety and decreased concentration and focus. This had an overall impact on the time management of the project as it often delayed progression leading to falling behind on deadlines previously set. This only started to get better at the beginning of April 2024.

### 4.16 Successes & Research progress

#### 4.16.1 Resilience

Despite these hurdles, over the course of the project, progress was positive but addressing specific issues such as vital networking issues will be crucial for future work in this domain. Throughout the course of this project, system hiccups have taught me a great deal of valuable lessons surrounding both hardware and software reliability, redundancy and volatility.

I believe this has increased my resiliency, allowing me to plan better for future projects/tasks with more knowledge than when beginning this originally. It has also reinforced my programming ability, learning and creating machine learning models within the space of 5 months alongside other brand new work.

#### 4.16.2 Research questions and Future improvement

In regards to the research questions asked at the beginning of the project, this project did slightly answer one of my questions. Could we use the federated learning for threat detection? I believe we could, yes. Federated learning provides data security which is much needed when working with highly sensitive data such as machine logs and network logs, the protection of data contained within these logs is essential.

This project did raise more questions however about the efficacy of the flower framework and scikit learn for this type of project. After multiple system failures and being left to play catchup, it's important to note here that there were setbacks in the project, however, it is clear the combination of flower and scikit learn for this type of project using LR and QRF may not be suitable, however this could also be combatted with some more uninterrupted research time to combat issues faced. Quantile regression forests were not fully supported by the scikit-learn package which was disappointing as I believe this would have allowed for much higher accuracy than logistic regression's predictions. The flower framework was also quite difficult to work with, fiddly at times until it was updated. An example of this being creating a custom implementation rather than using an out of the box solution, as was done for this project. This produced many errors that seemed nearly endless compared to if this was carried out using basic machine learning it may not have been the case.

It would be most important to see future work implement the visual user interfaces such as TheHive and the SIEM interface so malicious events can be alerted on and investigated further. It would be nice to see at least one other high value dataset such as the SOREL-20M dataset implemented across multiple clients. This particular dataset has 20 million records consisting of over eight terabytes of data which can easily be spread across multiple clients for training to simulate an environment such as multiple companies participating in a federated learning network and may help towards negating any potential overfitting issues encountered in this project.

Further development would also involve potentially moving the project to a better and more developed albeit more complex federated learning environment. Although Flower is a good environment to work with, it seems to be limited due to it still being quite new and mainly reliant upon being updated by the

community and developers from time to time. Other frameworks like tensor flow federated (TFF) made by google or PySyft which is similar to flower but has had much more time to develop and is backed by companies such as meta and google would be a much better choice for this project.

Mojo might also be interesting to try and implement as one issue faced throughout this project was the speed of using the federated learning, loading clients and the server was slower than desired. (Cooper, n.d.) Mojo is a programming language that is developed specifically for the use of AI and is deemed to be 90,000 times faster than python but works in an incredibly similar way by implementing both python syntax and ecosystem.

#### 4.16.3 Personal Development

Throughout the duration of the project, I also learned a considerable amount of knowledge surrounding various areas: machine learning, federated learning, python, virtual environments, data logging, networking etcetera. In the beginning I had very little knowledge of python, with basics learned from first year of university and a refresher of this when working with data at my placement in a SOC. Previously I have never worked on any machine learning code or projects so had not been exposed to many of the tools used for dataset manipulation, federated learning or graph plotting.

The main reason why this project was chosen was as result of working in a SOC environment where many FP alerts were received causing a significant amount of disruption frequently despite alert logic being in place to prevent this. With logic in place for many alerts adding exceptions for expected activity and certain regex matches, it was clear that fields which were not being utilised could potentially be used for better matching on TP matches and preventing FP with the use of machine learning, or with the use of federated learning, localising the machine learning to each machine to counter ML's main flaw which is secure data being sent across the network.

More personally, the project along with its shortcomings has improved my resiliency to setbacks and failure, enhancing my previous skills such as thinking on the fly, stress management and the ability to dynamically adapt.

As I have also never carried out a project of this calibre, any content I have previously made has always been for personal use meaning ethical considerations had not been thought of since there would be no impact to any third party. Although this rang true for this project also, the requirement to complete the ethics form Appendix D: Ethics Form provided me with personal development expanding my awareness of ethical considerations to be reviewed when carrying out future work and/or projects.

## 5 References

- Allwright, S. (2022). *What is a good accuracy score? Simply explained*. <https://stephenallwright.com/good-accuracy-score/>
- Asad, M., Moustafa, A., & Ito, T. (2021). *Federated Learning Versus Classical Machine Learning: A Convergence Comparison*.
- Augusto Remillano II, & Jemimah Molina. (2020). *New Mirai Variant Expands, Exploits CVE-2020-1017*. [https://www.trendmicro.com/en\\_us/research/20/g/new-mirai-variant-expands-arsenal-exploits-cve-2020-10173.html](https://www.trendmicro.com/en_us/research/20/g/new-mirai-variant-expands-arsenal-exploits-cve-2020-10173.html)
- Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K. H., Parcollet, T., Porto, P., De Gusmão, B., & Lane, N. D. (2020). *Flower: A Friendly Federated Learning Research Framework*. <https://arxiv.org/abs/2007.14390v5>
- Bhavitha, B. K., Rodrigues, A. P., & Chiplunkar, N. N. (2017). Comparative study of machine learning techniques in sentimental analysis. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICT 2017*, 216–221. <https://doi.org/10.1109/ICICT.2017.7975191>
- Breiman, L. (2001). *Random Forests*. 45, 5–32.
- Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Müller, A. C., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). *API design for machine learning software: experiences from the scikit-learn project*. <https://arxiv.org/abs/1309.0238v1>
- Bukhari, S. M. S., Zafar, M. H., Houran, M. A., Moosavi, S. K. R., Mansoor, M., Muaaz, M., & Sanfilippo, F. (2024). Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with SCNN-Bi-LSTM for enhanced reliability. *Ad Hoc Networks*, 155, 103407. <https://doi.org/https://doi.org/10.1016/j.adhoc.2024.103407>
- Campos, E. M., Saura, P. F., González-Vidal, A., Hernández-Ramos, J. L., Bernabé, J. B., Baldini, G., & Skarmeta, A. (2022). Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges. *Computer Networks*, 203, 108661. <https://doi.org/10.1016/J.COMNET.2021.108661>
- CentOS7. (n.d.). Retrieved May 3, 2024, from <https://www.centos.org/download/>
- Christopher, A. (2021). *K-Nearest Neighbor. A complete explanation of K-NN | by Antony Christopher | The Startup | Medium*. The Startup. <https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4>
- Cleophas, T. J., & Zwinderman, A. H. (2021). Quantile Regression. *Regression Analysis in Medical Research*, 453–467. [https://doi.org/10.1007/978-3-030-61394-5\\_27](https://doi.org/10.1007/978-3-030-61394-5_27)
- Committee on National Security Systems (CNSS) Glossary. (2022). <https://www.cnss.gov/CNSS/openDoc.cfm?a=HLim6uEMQo%2B44%2FT67ohq7w%3D%3D&b=84807CC6134B596C7F725C6F8A37C8581DB796F576DE1D96F96BFA079A4697D0CB0CE53FF94299B3ECC1925E2A6A5713>
- Cooper, D. (n.d.). *Mojo, 90,000 Times Faster Than Python, Finally Open Sourced! Just Launched, Already Surpassing 17,000 Stars | by Dylan Cooper | Apr, 2024 | Stackademic*. Retrieved May 3, 2024, from

- <https://blog.stackademic.com/mojo-90-000-times-faster-than-python-finally-open-sourced-777bdd9a1896>
- Cyber security breaches survey 2023 - GOV.UK.* (2023). <https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023>
- Deitterich, T. G. (2002). *Ensemble Learning*.
- Evangelou, M., & Adams, N. M. (2020). An anomaly detection framework for cyber-security data. *Computers & Security*, 97, 101941. <https://doi.org/10.1016/J.COSE.2020.101941>
- Gayhardt, L. (2023). Machine Learning Algorithm Cheat Sheet for Azure Machine Learning designer. In *Azure Machine Learning | Microsoft Docs* (p. 1).
- Joshi, N. (2022). *4 Key Differences between Federated Learning and Classical Machine Learning*. <https://www.linkedin.com/pulse/4-key-differences-between-federated-learning-classical-naveen-joshi/>
- Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., ... Zhao, S. (2019). Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning*, 14(1–2), 1–210. <https://doi.org/10.1561/22000000083>
- Konečný, J., Brendan McMahan, H., Yu, F. X., Theertha Suresh, A., Bacon Google, D., & Richtárik, P. (2016). *Federated Learning: Strategies for Improving Communication Efficiency*. <https://arxiv.org/abs/1610.05492v2>
- Kufel, J., Bargieł-Łączek, K., Kocot, S., Koźlik, M., Bartnikowska, W., Janik, M., Czogalik, Ł., Dudek, P., Magiera, M., Lis, A., Paszkiewicz, I., Nawrat, Z., Cebula, M., & Gruszczńska, K. (2023). What Is Machine Learning, Artificial Neural Networks and Deep Learning?—Examples of Practical Applications in Medicine. *Diagnostics*, 13(15), 2582. <https://doi.org/10.3390/diagnostics13152582>
- Kwon, D., Natarajan, K., Suh, S. C., Kim, H., & Kim, J. (2018). An empirical study on network anomaly detection using convolutional neural networks. *Proceedings - International Conference on Distributed Computing Systems, 2018-July*, 1595–1598. <https://doi.org/10.1109/ICDCS.2018.00178>
- Linux, the Operating System of Choice.* (2019). <https://ivypanda.com/essays/linux-the-operating-system-of-choice-research-paper/>
- Liu, B., Lv, N., Guo, Y., & Li, Y. (2023). *Recent Advances on Federated Learning: A Systematic Survey*. <https://doi.org/10.1145/nnnnnnnn.nnnnnnn>
- Luchs, M., & Doerr, C. (2019). *The Curious Case of Port 0*.
- Moustafa, N., Creech, G., & Slay, J. (2017). *Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models*. 127–156. [https://doi.org/10.1007/978-3-319-59439-2\\_5](https://doi.org/10.1007/978-3-319-59439-2_5)
- Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings*. <https://doi.org/10.1109/MILCIS.2015.7348942>

- Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1–3), 18–31. <https://doi.org/10.1080/19393555.2015.1125974>
- Moustafa, N., Slay, J., & Creech, G. (2019). 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. <https://doi.org/10.1109/TBDATA.2017.2715166>
- Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot and Édouardand, M., Duchesnay, and Édouard, & Duchesnay EDOUARDDUCHESNAY, Fré. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85), 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- Pissanidis, D. L., & Demertzis, K. (2024). *Integrating AI/ML in Cybersecurity: An Analysis of Open XDR Technology and its Application in Intrusion Detection and System Log Management*. <https://doi.org/10.20944/PREPRINTS202312.0205.V2>
- Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. K. A. A. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Computer Science*, 171, 1251–1260. <https://doi.org/10.1016/J.PROCS.2020.04.133>
- Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2021). NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 371 LNICST, 117–135. [https://doi.org/10.1007/978-3-030-72802-1\\_9](https://doi.org/10.1007/978-3-030-72802-1_9)
- Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2023). Cyber Threat Intelligence Sharing Scheme Based on Federated Learning for Network Intrusion Detection. *Journal of Network and Systems Management*, 31(1), 1–23. <https://doi.org/10.1007/S10922-022-09691-3/FIGURES/5>
- Shanmugarasa, Y., Paik, H. young, Kanhere, S. S., & Zhu, L. (2023). A systematic review of federated learning from clients' perspective: challenges and solutions. *Artificial Intelligence Review* 2023 56:2, 56(2), 1773–1827. <https://doi.org/10.1007/S10462-023-10563-8>
- sklearn.preprocessing.OneHotEncoder* — *scikit-learn 1.4.2 documentation*. (n.d.). Retrieved May 3, 2024, from <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>
- sklearn.preprocessing.RobustScaler* — *scikit-learn 1.4.2 documentation*. (n.d.). Retrieved May 3, 2024, from <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>
- Sodhi, P., Awasthi, N., & Sharma, V. (1354). *Introduction to Machine Learning and Its Basic Application in Python*. <https://ssrn.com/abstract=3323796>
- Ubuntu Releases*. (n.d.). Retrieved May 3, 2024, from [https://releases.ubuntu.com/?\\_gl=1\\*19oywzx\\*\\_gcl\\_au\\*MTY0Nzg2MzQ1Ny4xNzE0MzQ3MTk5&\\_ga=2.118175369.1657325220.1714735287-610647850.1714735287](https://releases.ubuntu.com/?_gl=1*19oywzx*_gcl_au*MTY0Nzg2MzQ1Ny4xNzE0MzQ3MTk5&_ga=2.118175369.1657325220.1714735287-610647850.1714735287)
- Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., & Zhang, W. (2023). A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2), 513–535. <https://doi.org/10.1007/S13042-022-01647-Y/TABLES/3>



## 6 Appendices

### 6.1 Appendix A: PID

Project Initiation Document

#### 1. Basic details

<b>Student Name</b>	Adam Cooper
<b>Draft Project Title</b>	Artificially Intelligent SIEM for Targeted Threat Detection and Analysis
<b>Course</b>	Cyber Security and Forensic Computing
<b>Project Supervisor</b>	Rahim Taheri

#### 2. Degree suitability

This project dives into the application of an artificially intelligent Security Information and Event Management (SIEM) system. How artificial intelligence, more specifically machine learning, can be applied to a SIEM and will build on prior constructs from the Cyber Security and Forensic Computing course with the usage of Python, virtual machines, networking and identification of vulnerabilities through forensic investigation.

### 3. The project environment and problem to be solved

The main audience of this project/report is cyber security analysts and those within security operation centres. These will be the main users of SIEM tools alongside threat detection and analysis tools that will be used daily to prevent attacks on the estate.

This research should provide a closer look into AI tooling for SIEM's, specifically targeting threat detection and analysis to prevent noise, false positives and increase alerting infrastructure by providing a more in-depth alerting. Currently SIEM's such as Splunk and Apache Metron, which do not have any AI implemented, lack this and face the following issues:

- Alert Noise – Due to poorly setup or bad alerting
- False positives – Poor alerting based on rules and not other activity
- Alerts allowlisted not on a granular level – Activity excepted based on little common factors
- Lack of consistency (i.e. alerts aren't updated in time) – Rules have to be manually created and maintained

Currently, AI in Cyber is still a niche sector with very few companies engaging in and implementing AI into their products. Similarly, and even more niche, AI used for threat detection and analysis with Artificially intelligent SIEM's are only being provided by a handful of companies.

Some of the most recognisable being Elastic, Darktrace, Exabeam and Solarwinds.

#### Research Question

Can AI effectively and accurately reduce alert noise from false positive alerting and similar previous activity whilst providing a more granular overview to alerting by implementing machine learning into a SIEM?

## 4. Project aim and objectives

**Aim:** To successfully implement AI into a SIEM for increased efficiency of alerting and reducing alert noise

### Objectives:

- Complete a review of previous works on AI SIEM implementations and machine learning for anomaly detection.
- Create virtual SOC environment to produce realistic results.
- Create Machine learning model.
- Train and test against previous datasets.
- Implement machine learning into SIEM to test against logs from SOC Environment
- Validate and Evaluation of results.
- Submit a project which has had an impact on the research aim and answered research questions.

## 5. Project constraints

- Project timescale.
- Possibility of GDPR compliance issues.
- BCS code of conduct..
- University code of conduct
- Personal hardware limitations.

## 6. Facilities and resources

Custom Desktop specification of:

- Ryzen 5 3600 6-core (12 CPU)
- AMD 5600XT 8GB
- 48GB 3600
- Personal Laptop specification of:
  - I7 8700k
  - Quadro p2000
  - 32gb 2133
- VMware 17x – For creating the SOC Environment and collecting logs.
- VSCode – IDE for interacting with playbooks.
- Python 3.11x – Used for coding ML Algorithms.
- Microsoft 365 – Word processing software.
- Jira – Project Management.
- Supervisor – Mentoring & Advanced knowledge on subject.
- Google Scholar – Research Papers.
- IEEXplore – Research Papers.
- Internet – Research.

## 7. Log of risks

No	Description	Likelihood (high, medium, low)	Impact	Mitigation/Avoidance
1.	Data Loss	Medium	Delay to project will have to regather data	Once data is collected create a backup which is to be destroyed at the end of the project.
2.	Total Hardware Failure	Low	Severe delays to project timeline and potentially impact performance – Loss of data and SOC Environment	Code backed up to GitHub private repository and work backed up to OneDrive so this can be accessed elsewhere if needed
3.	Laptop or Desktop Failure	Medium	Delay project and data loss	Create backups of work USB and external sources i.e. OneDrive/GitHub
4.	Missed Deadline	Medium	Sets back project	Ensure deadlines are strictly kept to completing stages of project in advance
5.	Corrupted Datasets	Low	Cause training data to become invalid or unusable	Create a backup of the superset dataset and backup trained sets.
6.	Corrupted Virtual Machines	Low	Partial loss of SOC environment and logging/data ingest	Frequently take snapshots which will be backed up to both my laptop and desktop
7.	Incorrect ML Algo	Medium	Setback project timeframe	Ensure algorithms used are correct for purpose
8.	Inability to access internet	Medium	Unable to carry out research on machine learning algorithms, techniques or ingest data for logging a realistic scenario	I will have the option to use a backup connection should this problem occur.

## 8. Project deliverables

Artefacts which will be created as part of the project include a SOC environment to create a realistic attack scenario and provide as accurate as possible logs within the timeframe provided. Using a pre-made SIEM application I aim to ingest these logs and implement machine learning model trained on a superset of previous datasets aggregating both machine and network activity.

My report will include a variety of documentation including previous works within the field, testing, system design, code along with future improvements and implementations.

## 9. Project approach

Project management and delivery will use Agile and be maintained through Jira. Secondary research will be carried out in the form of a literature review covering machine learning for anomaly detection and artificially intelligent SIEMs. As the subject of AI is relatively new to me, I will also be learning this throughout the duration of the project by referencing official documentation.

In addition, I will also:

- Complete a review of machine learning models with what would be best suited for the project i.e., researching different algorithms for anomaly detection such as KNN and Decision trees.
- Build and develop a virtual environment, similar to a SOC for gathering attack data logs to simulate a real-life scenario for threat intelligence.
- Create machine learning model(s) and deploy against test data (aggregated datasets) and logs gathered from test environment.
- Implement system and model validation to confirm the model detected threats as expected and did not need altering.
- Evaluate both system and model to ensure they successfully fulfilled their requirements and explore potential improvements.

## 10. Project Tasks and Timescales

No	Stage	Dates	Main Tasks
1.	Project Start Up	18/09/23 - 06/10/23	Decide on project and find supervisor
2.	PID	06/10/23 – 20/10/23	Complete PID
3.	Build Linux Machines	16/10/23 – 20/10/23	Create base virtual machines in VMWare 17
4.	Build windows Machines	20/10/23 – 24/10/23	Create base windows machines in VMWare 17
5.	Install T-POT	25/10/23 – 26/10/23	Install T-Pot on ubuntu server to act as a vulnerable access point within the network
6.	Install SIEM	26/10/23 – 29/10/23	Install SIEM application which can have machine learning implemented
7.	Aggregate Datasets	29/10/23 – 03/11/23	Aggregate multiple datasets to create a superset that can be used for training a machine learning model
8.	Complete building SOC Environment	20/10/23 – 20/11/23	Complete building SOC environment – Finish implementation, testing etc.
9.	Literature Review	20/10/23 – 11/11/23	Complete Secondary research – Have around 50-100+ articles of literature and research questions to build the project on
10	Build ML Models	12/11/23 – 20/12/23	Based on secondary research create machine learning model(s)
11	Train and Test models	12/11/23 – 20/12/23	Train and test machine learning model based on the superset of data
12	Implement model into SIEM	12/11/23 – 20/12/23	Implement these models into the SIEM to label incoming data
13	Gather logs to produce data	21/12/23 – 12/01/24	Open up network to conduct primary research gathering adversary attack data on the network. Carried out within a demilitarized zone (DMZ)
14	Dissertation Write-Up	01/01/24 – 30/04/24	Begin creating structure, formalising anything which has already been written, i.e., literature review and writing dissertation based on

			secondary and primary research
15	Continued Testing	01/01/24 – 30/04/24	Continue to test against logs or re-run against new log data which has been collected
16	Continued Research	01/01/24 – 30/04/24	Continue to research
17	Project submission	01/04/24 – 30/04/23	<p>Finalise writeup during this month – Reading through ensuring everything makes sense and everything that is needed has been included:</p> <ul style="list-style-type: none"> <li>• Graphs</li> <li>• Statistics</li> <li>• Citations</li> <li>• Appendices</li> <li>• ...etc.</li> </ul>

## 11. Supervisor Meetings

Timeframe	Description	Medium
Bi-Weekly	If anything, notable to be discussed or anything needs to be discussed regarding the project progression.	Email – In Person – Zoom

## 12. Legal, ethical, professional, social issues

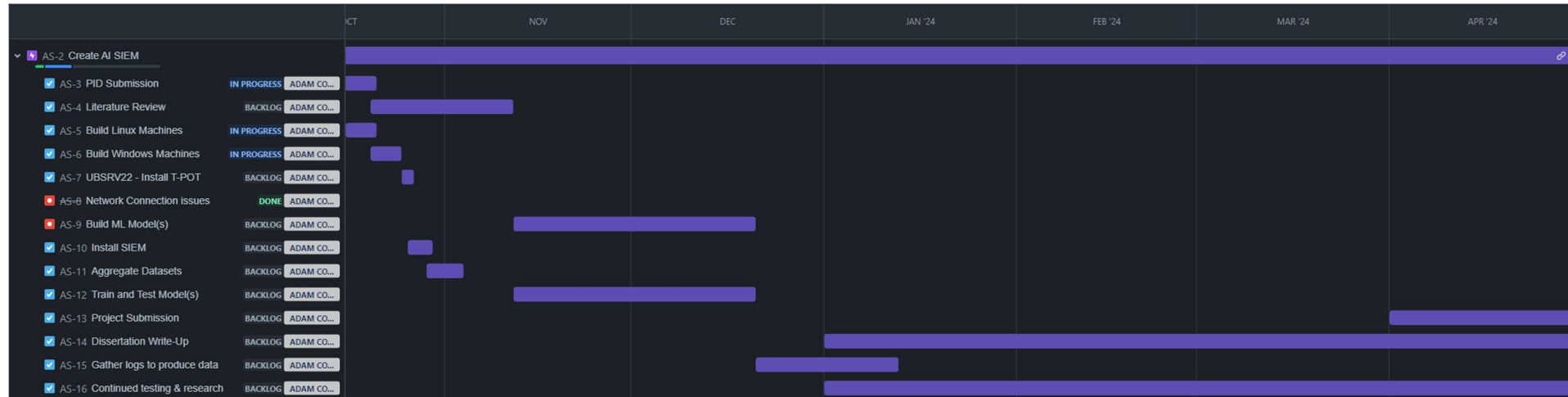
Any work carried out within this project is my own work and tools that are used throughout will be opensource with their sources stated and compliance to their conducts along with GDPR to ensure that no PII is being processed, the university code of conduct and BCS code of conduct. Any data collected is to be destroyed at the end of the project.

There will be no human participants as part of this project, the network will be open to the public and therefore adversaries can engage and attack the network at their own will. The network will be separated from any other network, running in a demilitarized zone and all systems are run from within virtual machines.

There will be no financial inducements or risks to physical or environmental features, historical or cultural heritage. The project does not involve animals and will not be harmful to any third parties nor can it affect UK security.



## 6.2 Appendix B: Gantt chart



### 6.3 Appendix C: Code

[HomeLab/ML Dissertation/ADAMFL/LogReg at master · sobraxis/HomeLab \(github.com\)](#)

## 6.4 Appendix D: Ethics Form



# Certificate of Ethics Review

Project title: Artificially Intelligent SIEM for Targeted threat detection and analysis

Name: Adam Cooper      User ID: 2009045      Application date: 18/04/2024 11:20:00      ER Number: TETHIC-2024-108463

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are [Elisavet Andrikopoulou](#), [Kirsten Smith](#)

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- [University Policy](#)
- [Safety on Geological Fieldwork](#)

It is also your responsibility to follow University guidance on Data Protection Policy:

- [General guidance for all data protection issues](#)
- [University Data Protection Policy](#)

Which school/department do you belong to?: **School of Computing**

What is your primary role at the University?: **Undergraduate Student**

What is the name of the member of staff who is responsible for supervising your project?: **Hamidreza Khaleghzadeh**

Is the study likely to involve human subjects (observation) or participants?: No

Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No

Are there risks of significant damage to physical and/or ecological environmental features?: No

Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No

Does the project involve animals in any way?: No

Could the research outputs potentially be harmful to third parties?: No

Could your research/artefact be adapted and be misused?: No

Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

Please read and confirm that you agree with the following statements: I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc.), I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs, I confirm that I will act ethically and honestly throughout this project

### Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments:

Supervisor's Digital Signature: **hamidreza.khaleghzadeh@port.ac.uk**      Date: **21/04/2024**