School of Computing, Engineering and Digital Technologies

Department of Computing and Games

Teesside University

Middlesbrough TS1 3BA

<u>Computing Masters Project - CIS4055</u>

AI-Driven Alert Prioritization & Fatigue Reduction Framework for SOCs

(Analysis, Design, and Implementation Report)

Student Name: Hiteesh Pushpendra Kondepati

Student ID: S3021074

Course: MSc. Computer Science

Submission Date: 06/01/2026

Module Leader: Dr. Zia Ush Shamszaman

Module Supervisor: Hannah Stothards

Total words Count-12,137

Acknowledgement

I am pleased to say that I am greatly indebted to my supervisor who gave me guidance, appropriate feedback and academic advice throughout the duration of this research project.

I would like to acknowledge the faculty members of the school of computing, engineering and digital technologies at Teesside University, who granted me the academic background and the resources required to complete this study.

I would also like to give credit to the developers and the maintainers of publicly available cybersecurity datasets, (UNSW-NB15 and CICIDS2017), that enabled the experimental evaluation in the current research.

I would like to thank the authors and researchers whose earlier work was the foundation of the literature review and enabled to create the direction of this study.

Declaration

I am hereby confirming that the dissertation entitled AI-Driven Alert Prioritization and Fatigue Reduction Framework of SOCs is my original work that has not been presented at this or any other institution to earn any other degree or qualification.

In this study, all the sources of information, data and literature have been duly considered and cited as advised by the provisions of academic honesty.

The study contained in this dissertation adheres to ethical principles and practices of Teesside University.

I testify that the work was executed under the guidance of the designated supervisor and did not only demonstrate my personal knowledge, evaluation, and perception of the topic.

Name: Hiteesh Pushpendra Kondepati

Student ID: S3021074

Date: 06/01/2026

Contents

Figures

Abstract

There is an incessant stream of security alerts at Security Operations Centres (SOCs), most of which are affected by false alarms causing analyst burnout and false alarms being missed. The proposed project is devoted to creation and testing of various machine learning classification models to automatically classify and rank security alerts by their probability to reflect a genuine threat. The paper aims to process alert data, design features of interest, and compare the results of various classifiers Logistic Regression, Random Forest, XGBoost, Support Vector Machine (SVM), and Neural Networks using benchmark cybersecurity datasets including UNSW-NB15 and CICIDS2017. All models will be evaluated based on their capacity to differentiate effectively between innocent and harmful alerts, and thus it will help minimize the number of false positives in SOC settings.

The analysis will be performed based on technical and practical effectiveness where metrics like precision, recall, F1-score and ROC-AUC are evaluated to determine the best algorithm in terms of balancing the accuracy of detection and reduction of alerts. The importance of the features in the project will also be analysed and the model interpretability through explainability tools such as SHAP to make sure that the decision making is transparent. Finally, the study is expected to find the most efficient way of classifications to enhance the efficiency of the alert triage and thus assist the analysts work on the high-priority and high-risk incidents in the SOCs of the present era.

Chapter 1. Introduction

1.1 Overview

Organizations are today overly dependent on integrated systems, cloud-based environment, and real-time data flow to facilitate important processes in their operations. Issues such as this digital transformation has made it faster in terms of productivity and connectivity but has also made the attack surface of cyber threats grow exponentially. Security Operations Centres (SOCs) are the front-line defence and they continuously observe large volumes of security alerts created by intrusion detection, firewalls, endpoint monitoring systems and threat intelligence systems. These notifications are to indicate possible malicious activity, but due to the flood of such notifications (mostly a false positive) the alert fatigue has generated a big burden of operations.

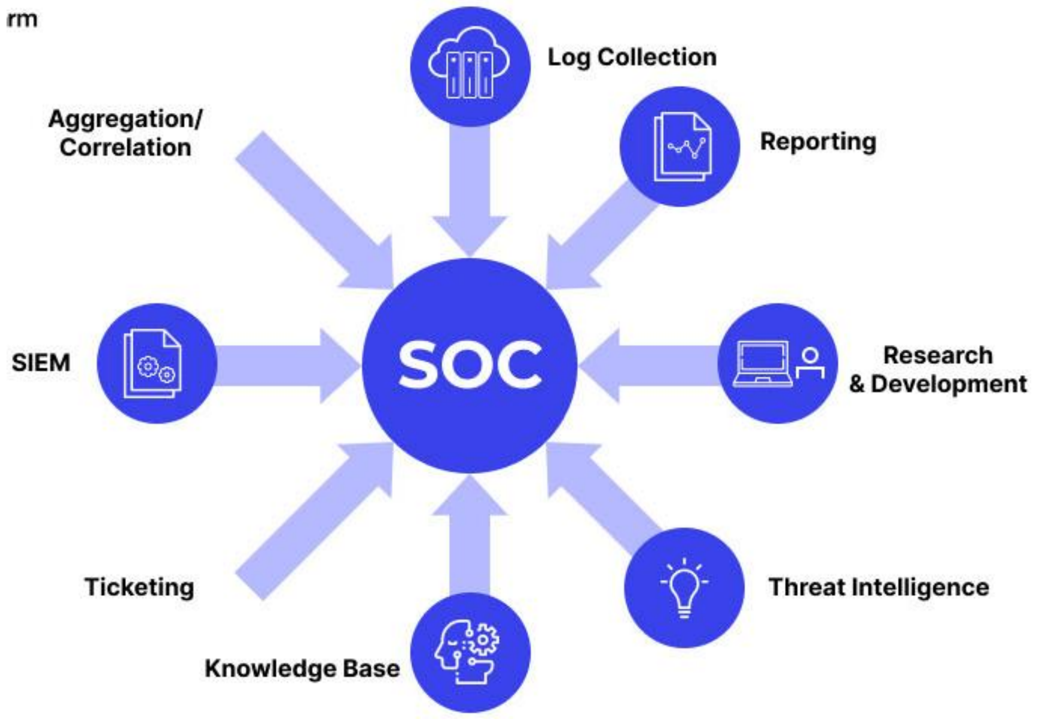


Figure 1. Main Functions of a SOC

[Image-source: https://www.wallarm.com/what/security-operations-center-soc]

An alert fatigue happens when the analyst becomes desensitized or overwhelmed with the signal of incoming alerts, which results in the failure to detect a real threat and slower incident responses.

Machine Learning (ML) offers potentially beneficial features of automating the classification of alerts. In contrast to the systems based on rules that rely on the predefined signatures, ML models develop based on the tendencies in data and respond to new types of attacks. The study uses publicly available datasets of cybersecurity (UNSW-NB15 and CICIDS2017) to apply, compare, and test a variety of classifiers such as Logistic Regression, Random Forest, XGBoost, Support Vector Machine (SVM), and Neural Networks. The project will attempt to offer empirical data on how well models perform, how easily they can be interpreted and how well they fit in actual SOC settings through stringent experimentation.

1.2 Background and Context

1.2.1 Essentiality

Digital security has ceased to be a supplementary business operation and now is an essential requirement in organizational survival. All organizations (finance, healthcare, government, and technology) are dependent on data integrity, system availability, and confidentiality to ensure that the trust and functionality are kept. Cybersecurity attacks may interrupt the workflow, reveal confidential information, and provoke serious financial and image losses. Thus, SOCs are crucial in that they monitor and react to suspicious activity that was made visible by security tools (SIEM) systems, IDS/IPS (Intrusion Detection/Prevention Systems) and Endpoint Detection and Response (EDR) platforms.

This difficulty outlines the necessity of automating the classification of alerts. Through machine learning, SOCs can rank alerts, which helps to reduce the load on analysts as well as make sure that the most serious threats are managed in time.

1.2.2 The Age of Data and Machine Learning

All systems, networks, and devices produce data which can be mined to discover latent data. In the case of cybersecurity, this data comprises network traffic patterns, system logs, and user behaviour measures all of which can be converted into features which can help detect an abnormal or malicious activity. The issue is not the availability of data, but the possibility to derive the meaning out of it both efficiently and accurately.

Machine Learning is a paradigm shift in data management in SOCs. Conventional rule-based strategies have drawbacks in the form of fixed definitions of attack. These systems have difficulty coping with the issue of zero-day exploits, polymorphic malware or insider attacks that do not conform to those patterns. But machine learning is able to make predictions based on historical data, find subtle correlations and detect emerging threats without being explicitly written.

In the meantime, explainable tools such as SHAP and LIME position automation between automation and trust to enable analysts to comprehend the reasons behind some alerts being interpreted as threats. Machine learning and cybersecurity synergy is therefore both a technological innovation and strategic need by contemporary organizations.

1.2.3 Problems and opportunities of Data-Driven Decision Making

Although the possibilities of ML in the field of cybersecurity are enormous, their application is fraught with a number of challenges. The first is data quality. Security datasets are commonly skewed, whereby there are many benign samples compared to malicious ones. Such imbalance may skew the models to forecasting safe results, which results in a missed detection. Also, real-world SOC data may be noisy, with redundant and incomplete and inconsistent features. To address these problems, preprocessing, feature engineering and resampling methods should be properly designed in order to guarantee reasonable and sound model training.

The second one is model interpretability. Decisions made under high-stakes situations such as cybersecurity need to be explainable. Analysts who will have to respond to an incident cannot depend on a black-box model that mark an alert as malicious without a reason. Thus, explainability techniques should be introduced to overcome the difference between automated systems and human knowledge.

The other problem is flexibility. The method of attacks changes fast and the models, which were trained on old information, may also become useless. Therefore, there must be continuous learning and retraining process. This leaves the paths to incremental learning and transfer learning, where the models also keep improving as the threat space is changing.

1.2.4 The Increasing Irrelevance of the Study

The increased applicability of AI and ML to cybersecurity has put smart alert classification as a research priority area. IBM 2023 Security Report indicates that organizations implementing AI-based monitoring systems cut down the time of detecting breaches by up to 40 percent

The research is especially topical as it aims at comparing several machine learning models to identify which of them are suitable to classify SOC alerts. The research does not present one model; rather, it critically analyses different methods of operation to offer practical information on the strengths, weaknesses, and trade-offs of every method. These comparative studies are essential to practitioners, because the model selection relies on issues such as data structure, interpretability needs and restrictions on computations.

Moreover, the research is in line with the popular trends in the industry that have focused on Explainable Artificial Intelligence (XAI). Since laws, such as the EU AI Act and the NIST guidelines, emphasize the transparency of AI systems, cybersecurity solutions should be interpretable as well as effective

1.3 Problem Statement

The overwhelming amount of security alerts represented by the various detection systems bomb Modern Security Operations Centres. Though these alerts are crucial in detecting the possible threats, the percentage of false positives and low severity is very high. This causes alert fatigue, a condition in which the analysts are overworked and hence respond slower, lose motivation and in most incidences, fail to notice actual attacks.

The complexity and scale of the modern cyber threats can no longer be dealt with using traditional systems with rules. They depend on pre-defined signatures and fixed thresholds, which are not capable of following the changing attack trends and situational peculiarities. Consequently, a gap between the quantity of noticed alerts and the ability to investigate them efficiently is expanding among security teams. The resultant effect is a paradox: there is an abundance of data like never before, but the capacity to take intelligent action on it has reduced.

Current AI-based solutions propose to solve this problem using alert correlation, prioritisation, or anomaly detection. Most studies, however, are either centred on one model or overlook interpretability and the analysts are unsure of the causes of automated decisions. A comparative study that will be used to evaluate various classification models based on their accuracy in addition to their realistic application and explanability in a SOC situation, is urgently required.

1.4 Research Question

The research question that will be used in this dissertation is:

Can I develop a unified, machine-learning-driven SOC framework that automates alert triage, integrates analyst learning, and predicts escalation likelihoods—while maintaining transparency and interpretability through explainable AI?

The question is an indication of both the technical and operations facets of the issue. It seeks to determine whether machine learning can differentiate benign and malicious alerts with an acceptable degree of accuracy, interpretability, and efficiency to make a more effective contribution to the actual SOC activities.

The secondary sub-questions comprise:

- What machine learning classifiers (e.g., Logistic Regression, Random Forest, XGBoost, SVM, Neural Network) are the best with regards to their performance in precision, recall, F1-score, and ROC-AUC on cybersecurity alert data?
- What is the effect of preprocessing and feature engineering process on the model performance and generalization ability?
- Is it possible to enhance the interpretability of model predictions to SOC analysts by using explainability tools like SHAP or LIME?
- What are the trade-offs between the accuracy of models, computation and interpretability of using ML-based alert classification systems?

All of these questions shape up the research design, experimentation and evaluation process all through the dissertation.

1.5 Aim, Approach and Objectives of the Study

1.5.1 Aim and Approach

This research does not only focus on comparing different machine learning (ML) models for alert classification but also is about creating a framework for an overall SOC automation system that includes alert triage, explainability, and human feedback components. The originality of this work is rooted in its hybrid approach that incorporates aspects of model explainability (through SHAP and LIME), sub-feature importance breakdown, and a theoretical ‘feedback loop’ process where human analysts can help in informing an iterative model revision cycle.

This study identifies the need for operational significance by changing from earlier research, which was satisfied with a purely algorithmic comparison, and examining how interpretable ML can be included in real-world SOC workflows.

The main aim of the study is to investigate, apply and test various machine learning classification models to classify security alerts aiming at determining which model yield most favourable results in terms of minimizing false positives and maximizing the accuracy of threat detection. It is about developing a powerful classification system that is able to handle massive amounts of cybersecurity data and assist in SOC operations by automating and explaining them.

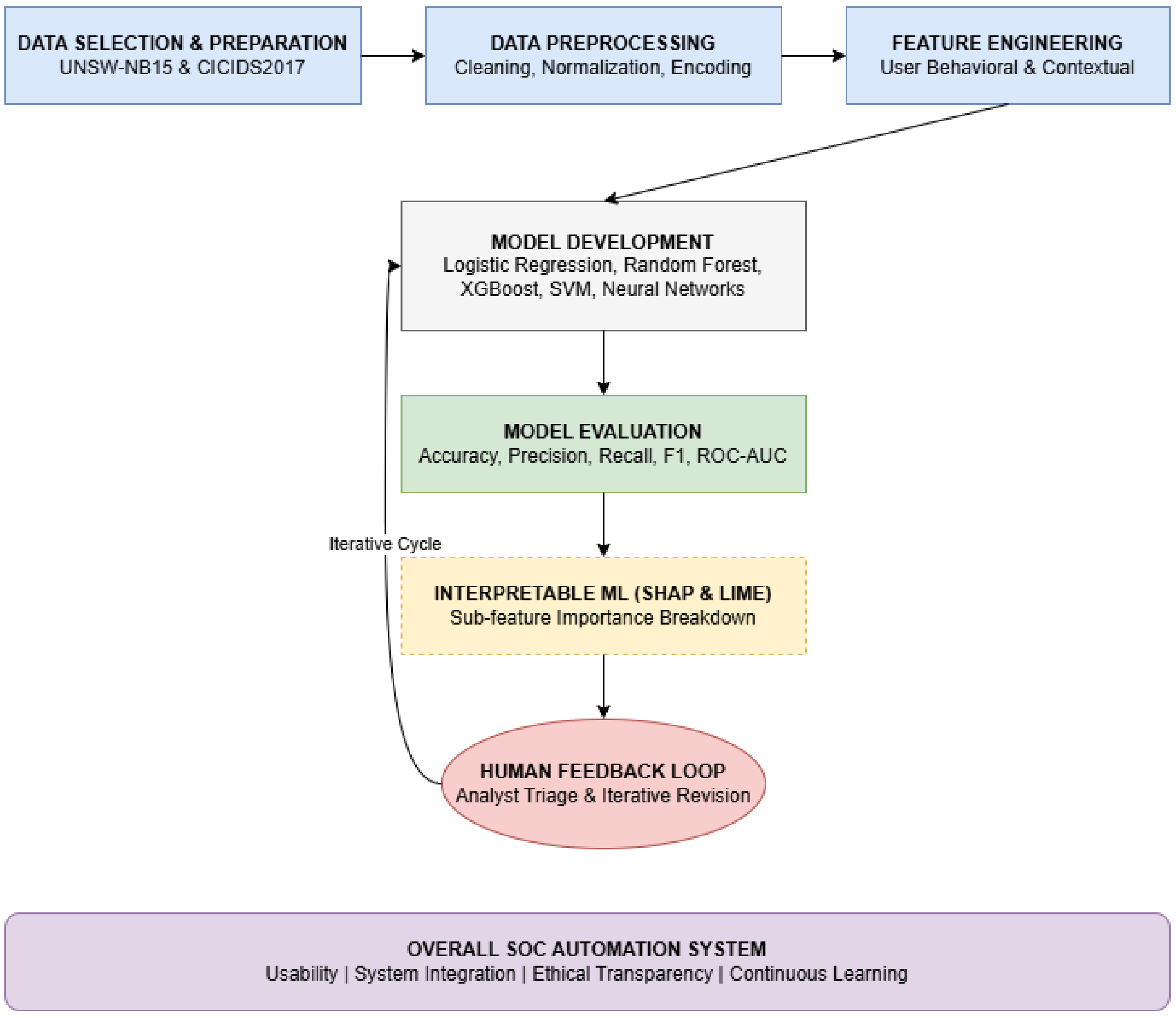

DATA SELECTION & PREPARATION
UNSW-NB15 & CICIDS2017
DATA PREPROCESSING
Cleaning, Normalization, Encoding
FEATURE ENGINEERING
User Behavioral & Contextual
MODEL DEVELOPMENT
Logistic Regression, Random Forest, XGBoost, SVM, Neural Networks
MODEL EVALUATION
Accuracy, Precision, Recall, F1, ROC-AUC
Iterative Cycle
INTERPRETABLE ML (SHAP & LIME)
Sub-feature Importance Breakdown
HUMAN FEEDBACK LOOP
Analyst Triage & Iterative Revision
OVERALL SOC AUTOMATION SYSTEM
Usability | System Integration | Ethical Transparency | Continuous Learning

Figure 2. Aim and Approach of the Study

The strategy consists of a number of steps:

- **Data Selection and Preparation:** The dataset used to do the experiment will be the UNSW-NB15 and CICIDS2017 databases, which consist of labelled network traffic of normal and malicious activity.
- **Data Preprocessing:** Removing and normalizing data, as well as encoding it to be processed by machine learning algorithms.
- **Feature Engineering:** The choice and extraction of usable features that user behavioural as well as contextual features of network traffic.
- **Model Development:** Model training and model testing of several classification models, including Logistic Regression, Random Forest, XGBoost, SVM, and Neural Networks.
- **Model Evaluation:** These metrics can be used to compare models based on accuracy, precision, recall, F1-score, and ROC-AUC and find the most successful classifier.

- **Performance Analysis:** Measuring the increase in the reduction of alerts, the rate of detecting alerts, and the interpretability of the model.

1.5.2 Research Objectives

Objective 1: To perform an in-depth analysis of the literature and studies regarding AI-based systems to manage alerts.

Objective 2: To prepare and preprocess cyberspace security data (UNSW-NB15 and CICIDS2017) to train and evaluate the model successfully.

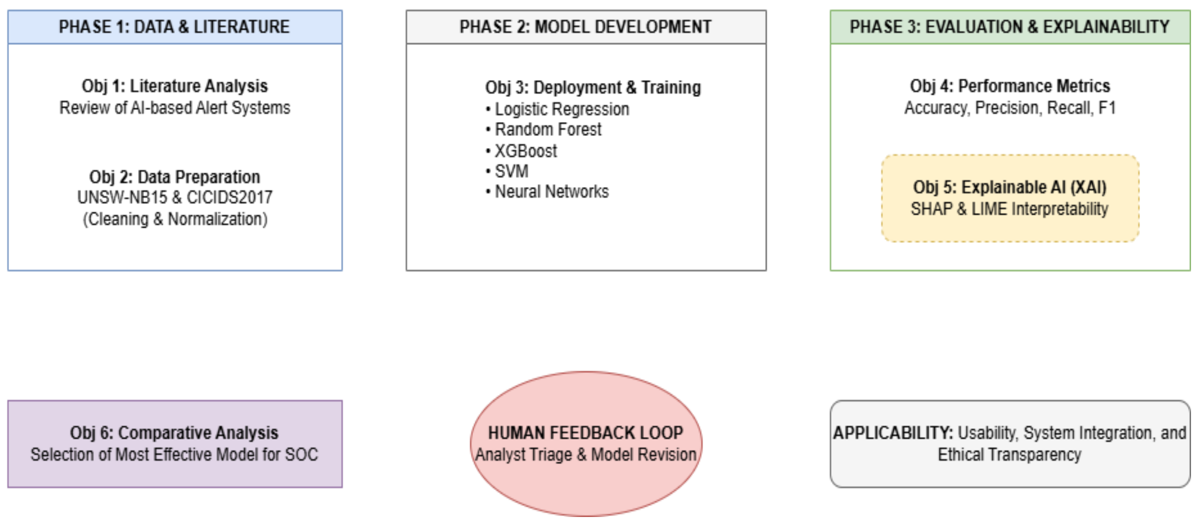

PHASE 1: DATA & LITERATURE
Obj 1: Literature Analysis
Review of AI-based Alert Systems
Obj 2: Data Preparation
UNSW-NB15 & CICIDS2017
(Cleaning & Normalization)
PHASE 2: MODEL DEVELOPMENT
Obj 3: Deployment & Training
• Logistic Regression
• Random Forest
• XGBoost
• SVM
• Neural Networks
PHASE 3: EVALUATION & EXPLAINABILITY
Obj 4: Performance Metrics
Accuracy, Precision, Recall, F1
Obj 5: Explainable AI (XAI)
SHAP & LIME Interpretability
Obj 6: Comparative Analysis
Selection of Most Effective Model for SOC
HUMAN FEEDBACK LOOP
Analyst Triage & Model Revision
APPLICABILITY: Usability, System Integration, and Ethical Transparency

Figure 3. Research Objectives

Objective 3: To deploy and train various machine learning classification models like the Logistic Regression, Random Forest, XGBoost, SVM, and Neural Networks in alert classification.

Objective 4: To compare and contrast the performance (Accuracy, Precision, Recall, F1-score, and ROC-AUC) of a model on a technical and practical basis.

Objective 5: To improve the understanding of interpretability by explainable AI methods using SHAP and LIME.

Objective 6: To introduce a comparative analysis and suggest the most effective machine learning model to classify alerts of SOC.

The study will be summarized by the identification of the most appropriate classifier that possesses the best balance between prediction capability, interpretability and computational efficiency. Quantitative evidence and qualitative interpretation will be used to support the recommendations.

1.6 Analytical Framework

The analytic framework integrates theoretical knowledge and empirical testing into a modular SOC automation framework which is intended to permit future integration into a single platform. The process of research has occurred in six main stages:

1. Data Collection
2. Preprocessing and Feature Engineering
3. Model Development
4. Model Evaluation
5. Interpretability and Explainability
6. Prototype Integration and Validation

Stages 1-5 follow a normative ML lifecycle, covering the stages of data preparation, training, and model evaluation relative to an alternative, traditional, learning model. Stage 6 includes a proof-of-concept prototype - a minimal viable product that simulates analyst engagement (practice) through an offline dashboard interface.

Although this initial version remains offline, the architecture is designed in modules, and intended to be integrated with SIEM or EDR systems via REST APIs in the future as part of the SOC environment. The prototype could be easily duplicated using lightweight tools developed in Python (Flask, Streamlit, etc.) to demonstrate alert scoring in real time, and capture analyst feedback to demonstrate how that process could work in a unified SOC process.

Stage 1: Data Acquisition

Two publicly accessible benchmark data sets, such as UNSW-NB15 and CICIDS2017, are used in the study and are commonly known in cybersecurity research. The datasets are various network behaviours that include normal traffic and various forms of cyberattacks, such as DoS, infiltration, and brute-force attacks. With them, it is possible to guarantee the reproducibility and the ethical standards because they are anonymised, and they have non-sensitive data.

Stage 2: Data Preprocessing and Feature Engineering

Raw security data usually include redundant, noisy or missing data. To address these challenges:

- Relevant features will be eliminated through data cleaning and null values will be processed.
- Categorical variables will be encoded to numerical form by encoding methods (e.g., one-hot encoding or label encoding).
- Normalization or standardization will transform numeric values to achieve optimal learning of a model.
- Class imbalance - another frequent problem of cybersecurity dataset - will be addressed through the use of some techniques like SMOTE (Synthetic Minority Over-sampling Technique) or weighting classes.

Stage 3: Model Development

Five machine learning models will be trained and optimized to be used in classification:

- **Logistic regression:** A interpretable binary classification baseline linear model.
- **Random Forest:** This is a powerful ensemble algorithm based on the integration of decision trees to enhance the accuracy and minimize overfitting.

- **XGBoost:** It is a gradient boosting algorithm, which is highly performant and has a high degree of scalability in structured data.
- **Support Vector Machine (SVM):** This is a model used to build the best hyperplanes to classify data, which works with high-dimensional spaces.
- **Artificial Neural Network (ANN):** It is a feed-forward network that has the ability to learn intricate non-linear associations between data.

The grid search and cross-validation will be used to tune each model with the aim of determining the best hyperparameters. The training will be performed with the help of standard Python-based ML frameworks: Scikit-learn and TensorFlow.

Stage 4: Model Evaluation

In order to provide the rigorous and fair comparison, several metrics will be used:

- **Accuracy:** Percentage of the correctly identified alerts.
- **Precision:** Ratio of threats that are correctly predicted to the total number of threats predicted (minimisation of false positives).
- **Recall:** Percentage of correctly identified threats to the overall false threats (minimisation of false negativity).
- **F1- Score:** Precision and recall have a harmonic mean, which is balance.
- **ROC-AUC:** The trade-off between false and true positive rates.

Stage 5: Model Interpretation and Explainability

In this work, I achieve explainability through the use of SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-Agnostic Explanations), to provide transparency into the model and build trust in the analyst.

In our case, since I didn't implement that functionality as part of this phase, I, instead, created a conceptual simulated analyst validation process—where I represented human judgment as part of controlled sequential review steps. Future design thinking includes a mechanism for continuous learning (i.e., corrections by an analyst would result in part of the retraining loop). The aforementioned envisions a degree of human–AI partnership and collaboration, which is a significant aspect awarded to intelligent SOC operations.

Stage 6: Validation, Discussion, and Practitioner Review

In addition to technical evaluation metrics (Accuracy, F1, and ROC-AUC), this project would highlight the importance of validating the practical utility and conduct external evaluations that would better indicate how the models would work in practice.

There are five intended or suggested validation activities to aid in practical assessment and usefulness and they are:

- Walkthroughs of the models by cybersecurity experts who may undertake the role of assessing its interpretability, alert ranking usability, and integration into analyst processes.

- Attempted survey or semi-structured interviews to the potential analysts to determine the usefulness and transparency of the system.
- Reviewing of design of the analytical framework by peers to ensure academic and operational validity.

1.7 Scope and Limitations

Offline experimentation is highlighted in this work to establish an evaluation baseline on which it can be controlled. In spite of the fact that models are tested using benchmark data, practical implementation poses challenges, including:

- Constant data flow and latency,
- Training models with the changing threat,
- Combination of SIEM (Security Information and Event Management) systems, and
- Administering access control and information privacy to a live running instance.

To deal with these issues, a high-level framework is suggested - based on containerized ML models (e.g., via Docker/Kubernetes, etc.) integrated with a SOC (Security Operations Centre) system, the ML models are accessed via an API to real-time inference/retraining/explainability dashboards.

1.7.1 Scope

This dissertation addresses the comparative efficacy of machine learning models for alert triage, albeit with diminished breadth compared to the originally envisioned study. The original idea was to include three synergistic components: alert classification, human-in-the-loop learning, and escalation prediction.

Although the planning for all three components remains, in this part of the study I only developed the alert classification component and the explainability component. These components will not be explored in this study for the reason expressed, given there was a limited amount of real SOC feedback data available, and method of escalation predictions was computationally intensive.

This dissertation is also limited to the classification of security alerts through various machine learning models in a controlled experimental setting. The project is oriented to offline testing with the help of the pre-existing benchmarking datasets instead of implementation in operational SOC systems.

Precisely, the study will:

- Focus on binary/ multi-class alert classification (malicious vs. benign, or by type of attack).
- Compare the performance of five chosen ML models, namely, Logistic Regression, Random Forest, XGBoost, SVM, and ANN.
- Adopt publicly available datasets (UNSW-NB15 and CICIDS2017) to make sure that it is reproducible and ethical.

- Add explainable AI technologies (SHAP and LIME) to interpretability.
- Pressure on interpretability of models, computational efficiency and scalability of the models are important comparative factors emphasized.

The results of the study will be recommended to be used in the real-life SOC operations, though the application of the results to the production setting is not within the short-term perspective.

1.7.2 Limitations

This study is aimed at discussing a difficult issue of classifying alerts. The contribution to threat escalation prediction, which was potentially to be made, of predicting which alerts would get moved into the high-severity category, was not obtained because no appropriate labelled escalation data was available. This has been recognized and supported because it is technically feasible and time-related complications.

Such extensions of the work in the future will consist of escalation modelling through a sequence-based process (i.e., LSTM or temporal boosting) as soon as longitudinal SOC data becomes accessible.

Although the study is designed in a structured way and has a sound methodology, it has a number of limitations:

- **Dataset Representativeness:** Although widely applicable, the UNSW-NB15 and CICIDS2017 datasets might not eloquently represent contemporary SOC data and its complexity and diversity. Real world data may contain unlabelled and dynamic and proprietary components which are not present in benchmark data.
- **Model Generalization:** The models will be trained and tested in the condition of stability. The performance of dynamic, evolving systems, e.g., continuous network monitoring can vary as the attack patterns and system configurations change.
- **Computational Constraints:** The more advanced models specifically neural networks consume a substantial amount of computational resources. Tuned hyperparameters with large scale training and high levels of hyperparameter tuning can be contained due to time and hardware constraints.

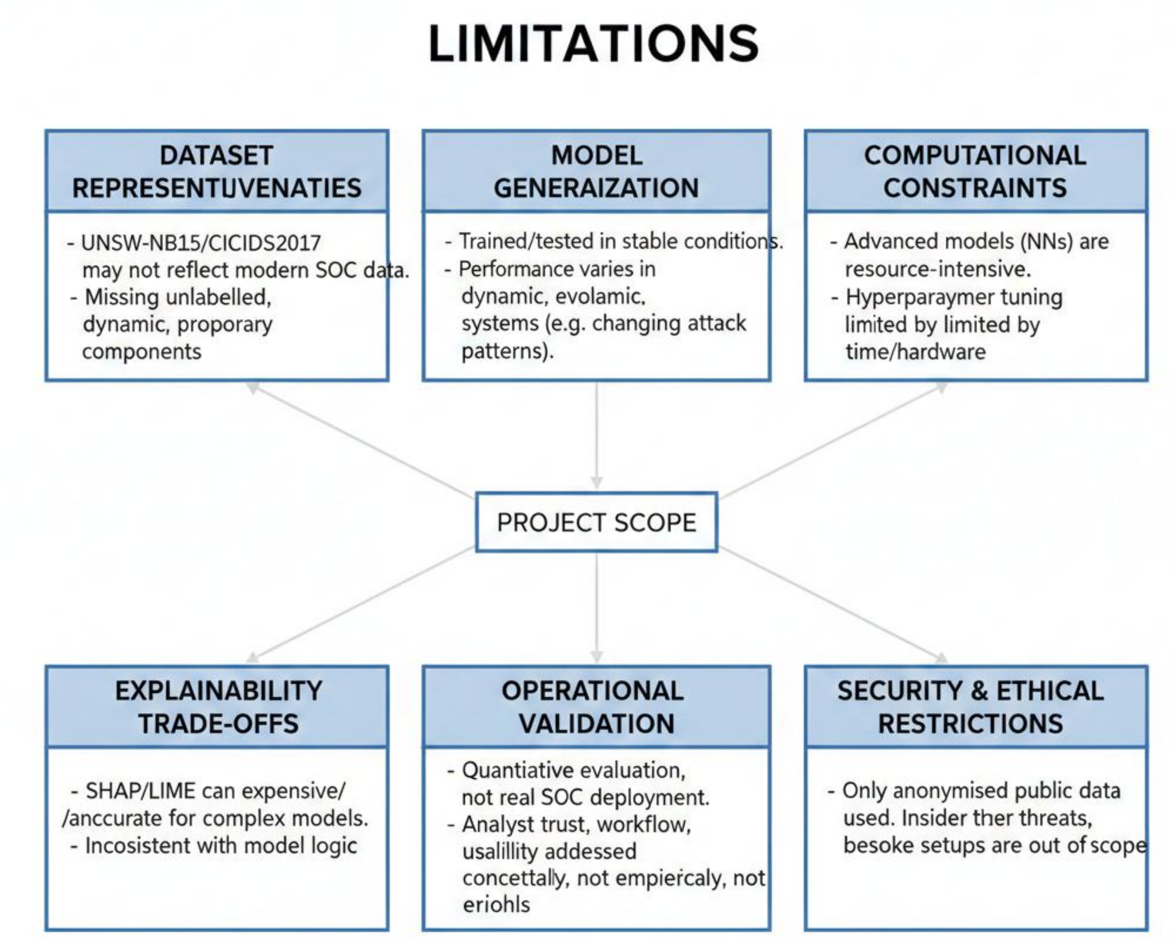

LIMITATIONS
DATASET REPRESENTUVENATIES
- UNSW-NB15/CICIDS2017 may not reflect modern SOC data.
- Missing unlabelled, dynamic, proporary components
MODEL GENERAIZATION
- Trained/tested in stable conditions.
- Performance varies in dynamic, evolamic, systems (e.g. changing attack patterns).
COMPUTATIONAL CONSTRAINTS
- Advanced models (NNs) are resource-intensive.
- Hyperparaymer tuning limited by limited by time/hardware
PROJECT SCOPE
EXPLAINABILITY TRADE-OFFS
- SHAP/LIME can expensive/ /anccurate for complex models.
- Incositent with model logic
OPERATIONAL VALIDATION
- Quantiative evaluation, not real SOC deployment.
- Analyst trust, workflow, usalillity addessed concettally, not empiercaly, not eriohls
SECURITY & ETHICAL RESTRICTIONS
- Only anonymised public data used. Insider ther threats, besoke setups are out of scope

Figure 4. Limitations

- **Explainability Trade-offs:** SHAP and LIME are interpretable, but when using complex models, each of these algorithms may be computationally expensive and sometimes inaccurate. The interpretations are not necessarily always entirely consistent with model logic.
- **Operational Validation:** The project is aimed at the quantitative evaluation and is not oriented to the deployment in the real SOC environment. Conceptually, therefore, such metrics as analyst trust, workflow integration, and usability will be addressed, as opposed to measuring them empirically.
- **Security Restrictions and Ethical:** Only anonymised publicly available datasets will be used in the research. As a result, certain practical undertones (e.g., insider threat or bespoke enterprise setup) are out of reach.

Nevertheless, the dissertation has a good scholarly input by coherently comparing machine learning models, offering insights on their interpretability, and proposing practical suggestions on SOC alert classification systems. The findings will be the basis of further study of real-time application, adaptive learning, and human-AI cooperation in cybersecurity.

1.8 Structure of the Report

- Chapter One includes an introduction that is a narrative presentation of the background, stating the context, the research question, the objectives, outlining the scope and providing a framework of the analytical approach.
- Chapter Two is a systematic review of available literature on machine learning use in cybersecurity, alert classification, and explainable artificial intelligence.
- Chapter Three builds up on the methodological framework, which entailed the selection of data set, preprocessing, development of the model, evaluation metrics, and the adopted analytical framework.
- Chapter Four reports the empirical results, benchmarking model, and summarizing key results based on the analyses.
- Chapter Five is the final chapter of the investigation as it summarizes salient findings, highlights contributions, reflects on limitations and suggests future research directions.

Chapter 2. Literature Review

2.1 Definition of Terms

- **Security Operations Centre (SOC):** The key contact point in the organization that constantly monitors, detects, analyses, and responds to cybersecurity incidents with the use of SIEM, IDS/IPS, and EDR systems, etc.
- **Security Alert:** An alert that is produced by a security monitoring system that there is a possible suspicious or malicious activity that needs to be investigated.
- **Alert Fatigue:** A situation in which SOC analysts face a problem of being overwhelmed by large amounts of alerts thus resulting in slow response times, lack of detection of threats and decrease in operational effectiveness.
- **False Positive:** An incident that is detected as a security threat by a detection system, that is benign or not malicious.
- **Alert Prioritization:** Ranking of alerts according to their perceived severity or probability of being a real security incident.

2.2 What Has Been Published in This Field

Past comparative research (Kaushik & Goel, 2021; Ma'aji et al., 2023; Malik & Singh, 2022; Naseer et al., 2018; Sharma, 2025) was used to establish benchmarks on the performance used to track the performance of ML-based intrusion detection and alert filtering. These papers demonstrate ensemble and deep-learning models to be highly accurate, yet opaque or, in other words, non-integrated to operations.

There is a recent shift to human centred SOC automation (Stiborek et al., 2023; Zhou et al., 2024). In these studies, human-in-the-loop frameworks are adopted to have analyst expertise in retraining models and active learning. Besides, Lin et al. (2023) and Alhaidari et al. (2024) consider the explanatory AI (XAI) tool usage among SOCs and mention possible advantages of transparency and enhanced trust in the analyst experience and system acceptance.

The challenges identified by Ogino, 2015 and Tang et al., 2016 regarding deployment include concept drift, latency, and continuous retraining (deployment challenges) and imply the necessity of flexible design systems. On the whole, the reviewed studies suggest SOC frameworks and a balance between accuracy, explainability, adaptability, and human-AI collaboration.

The article by **Kaushik and Goel (2021)** includes a comparison of machine-learning classifiers (Logistic Regression, SVM, Decision Tree, Random Forest) on the NSL-KDD network intrusion dataset, revealing the significance of feature selection and adequate pre-processing in achieving high detection rates. They discovered that ensemble techniques especially the Random Forest provides better results in terms of intrusion detection and false positive minimization, which is crucial in alert filtering in SOCs. The authors further note that a problem of imbalanced databases and redundancy in features is a continued problem in intrusion detection modelling. Their work can be used to defend the practical claim that monitored ML classifiers can be the basis of automated alert classifiers in practice in the security operations of the real world.

Ma'aji et al. (2023) conducts a comparative analysis of a number of machine-learning based intrusion detection models (such as Naive Bayes, Random Forest, AdaBoost) and shows that those based on ensemble can be more accurate and robust than simple classifiers. The study notes that most of the existing literature uses clean academic data sets and might not be very representative of the sloppy and changing nature of SOC alert data. Ma'aji et al., 2023 stresses the importance of models generalising well to real-world network traffic and variable attack types hence its compatibility with the purpose of the project which is to compare different classifiers to alert prioritisation.

Malik and Singh (2022) survey and experiment on supervised machine-learning methods to intrusion detection, demonstrating that even comparatively simple models can provide useful improvements overrule-based systems in the context of considered data preparation. Their exploration highlights that in most intrusion detection tasks, feature engineering and dataset quality are a bottleneck as opposed to model complexity per se. This supports your perspective about classification modelling (instead of complete incident escalation) in your project and the value of creating and testing a variety of models in similar conditions.

Naseer et al. (2018) discuss deep neural-network (DNN) algorithms, including CNNs, autoencoders and LSTMs, as an alternative to traditional ML classifiers, including Decision Trees, SVM and Random Forest, to detect anomalies in the NSL-KDD dataset. Their findings indicate that deep models are promising in terms of performance improvement (in AUC, precision-recall) in network anomaly detection, although they also introduce a high computational cost and large training requirements. This study offers a valuable comparison to the simpler classification models and advises the trade-offs that your project deserves to make between accuracy, interpretability and runtime.

Ogino (2015) considers machine learning as an intrusion detection tool in a real-time system environment (Jubatus framework), and identifies problems with model deployment, concept drift, and real-time issues. The paper demonstrates that in addition to the high offline classification accuracy, the implementation of many ML models in practice in a live SOC environment also features extra challenges, including processing latency, processing streaming data and constantly updating attack signature. This supports the importance of your study to the realistic assessment of classification models not only at offline measurements.

Tang et al. (2016) use deep-learning algorithms (when applied to the software-defined networking scenario) to intrusion detection and demonstrate that the temporal patterns in network traffic can be successfully represented by the LSTM-based models. According to their research, they propose that the exploitation of the sequence and timing of alerts can enhance the classification performance, and this can prove to be an interesting angle in future work in your project (although you are currently centring on classification) or hybrid modelling in your discussion on future directions.

There is a comparative analysis of machine-learning models (Random Forest, XGBoost, SVM and Neural Networks) to intrusion detection systems undertaken by **Sharma (2025)** where gradient-boosted models such as XGBoost tend to balance best accuracy, interpretability and computational efficiency. Another aspect that the paper highlights is that security application needs interpretability and trust particularly when the automated system is used to assist human analysts in the SOCs. This is closely related to your project focus on classification, learning based on analyst decisions and practical applicability of ML models.

S. No.	Citation	Main Concept	Remarks	Limitations
1	Kaushik & Goel (2021)	Comparative evaluation of ML classifiers for intrusion detection using NSL-KDD dataset	Ensemble models, particularly Random Forest, achieved higher detection rates and reduced false positives; feature selection and preprocessing are critical	Dataset imbalance and feature redundancy remain unresolved; based on benchmark dataset
2	Ma'aji et al. (2023)	Comparison of ensemble and simple ML models for intrusion detection	Ensemble-based models showed higher accuracy and robustness; emphasizes generalisation to real-world traffic	Uses mostly clean academic datasets, which may not reflect real SOC alert noise
3	Malik & Singh (2022)	Supervised ML methods for intrusion detection	Even simpler ML models outperform rule-based systems when data preparation is effective; highlights importance of feature engineering	Model performance heavily dependent on dataset quality rather than model complexity
4	Naseer et al. (2018)	Deep learning approaches for network anomaly detection	Deep models (CNNs, LSTMs, autoencoders) improve AUC and precision-recall metrics	High computational cost, large training requirements, and reduced interpretability
5	Ogino (2015)	Real-time intrusion detection using ML in SOC environments	Highlights deployment challenges such as concept drift, latency, and	Focuses on deployment issues rather than optimizing model accuracy

			streaming data handling	
6	Tang et al. (2016)	LSTM-based intrusion detection in software-defined networks	Temporal and sequential patterns improve detection performance	Focus limited to sequence-based models; higher computational complexity
7	Sharma (2025)	Comparative study of ML models for intrusion detection	XGBoost balances accuracy, interpretability, and computational efficiency; stresses analyst trust	Does not fully address long-term deployment or evolving threat adaptation

2.3 Uniqueness of the Study

- In contrast to most previous studies that consider the accuracy of intrusion detection alone, this study highlights the importance of alert prioritization and operational fatigue avoidance in SOC settings.
- The experiment takes a comparative multi model approach where the Logistic Regression, Randi Forest, XGBoost, SVM and Neural Networks are put through the same experimental conditions.
- It uses explainable AI (SHAP/LIME) as a part of the alert classification pipeline to overcome the ubiquitous limitation of black-box ML models in cybersecurity studies.
- The study suggests an idea of a human-in-the-loop structure, which will allow analyst input to inform the logic of prioritization, a factor that was frequently omitted in previous researches.
- This study is based on practical applicability of SOC, unlike pure theoretical work, which takes into account interpretability, analyst trust, and workflow integration in addition to technical metrics.
- Two current benchmark datasets (UNSW-NB15 and CICIDS2017) are used to further improve cross-dataset generalizability and empirical robustness.

2.4 Pros & Cons / Benefits & Challenges

2.4.1Benefits/Advantages

- **Reduction in Alert Volume:** Automated prioritization will go a long way to minimize low-value alerts and focus on high-risk incidences on the part of the analysts.
- **Increased Detection Accuracy:** Machine learning systems are better at detecting complex and changing patterns of attack than the traditional rule-based systems.

- **Scalability:** ML-based frameworks are also able to handle large amounts of security data more effectively compared to manual methods in triage.
- **Explainability and Trust:** XAI methods bring more confidence to the analyst, because they explain the underlying logic of prioritization of alerts in a transparent manner.
- **Operational Efficiency:** False positives reduce the SOC resources because they minimize the time they spend investigating such cases.

2.4.2 Challenges / Limitations

- **Data Imbalance:** The samples forming cybersecurity datasets have high proportions of non-malicious samples compared to malicious ones, which may tend to bias learning on models.
- **Trade-off of Model Interpretability:** Transparency may not be as much as simple models because the highly accurate models may be deep learning.
- **Concept Drift:** The patterns of attack change with time and one has to keep retraining the model to stay effective.
- **Computational Overhead:** Contemporary models and explainability methods may be resource-consuming.
- **Deployment Complexity:** Implementing ML models in the real-time SOC pipelines introduces issues of latency, governance, and security.

2.5 Research Gaps

- The majority of the literature has concentrated on intrusion detection performance, and little has been done on the performance of alerts and the reduction of the workload among the analysts.
- Comparative studies generally overlook the explainability, leaving a knowledge gap regarding the possibility to trust and justify the ML decisions by the SOC analysts.
- There is little literature that incorporates feedback mechanisms of the human-in-the-loop to adjust models on analyst expertise and with changing threats.
- Numerous research works are based on individual datasets or individual models, which limits the extrapolation to other SOC settings.
- The frameworks to integrate accuracy, interpretability and operational usability into one solution SOC-centric are lacking.
- Few papers directly compare alert fatigue as a quantifiable operational performance aspect, instead of accuracy as the only measure of success.

Chapter 3. Methodology

3.1 Research Design and System Overview

This thesis uses an applied experimental approach in constructing and testing a machine learning pipeline to triage alerts in a security operations centre with telemetry of high volume. The essence of the research is to transform raw records of flow level networks into an ordered queue of alerts that can aid in decision making by the analysts and minimize false positives that contribute to operational overload and fatigue (McGill et al., 2025; Kearney et al., 2023). The method incorporates a hybrid architecture that integrates supervised detection, temporal modelling, density-based clustering, and a human feedback rule to weight update which generates a bound priority score of 0 to 10 to each candidate alert. The incident response framing is consistent with the existing guidelines that underline the ability to conduct a consistent triage, analysis and prioritize the incident related data (National Institute of Standards and Technology, 2025).

The methodological design has four sequential stages. **First,** curated benchmark datasets are used to represent modern enterprise like traffic and attack behaviour. **Second,** a leakage controlled preprocessing pipeline generates model ready matrices with explicit separation of training and testing transformations. **Third,** supervised learners are trained for probabilistic attack likelihood estimation, with gradient boosted trees used for tabular discriminative learning and long short-term memory networks used for short horizon temporal pattern recognition (Chen, 2016; Hochreiter and Schmidhuber, 1997). **Fourth,** Unsupervised clustering is used to decrease alerts on similar events, maximizing downstream consolidation and escalation of rare or noisy patterns as noted by previous studies of alert clustering and prioritization (Ester et al., 1996; Shittu et al., 2015).

3.2 Datasets and Data Governance

Two publicly available intrusion detection datasets are employed so that the methodological reproducibility can be guaranteed and the models can be subjected to the heterogeneous feature spaces and traffic distributions. UNSW NB15 contains labelled network flow data that is modern in attack categories and has mixed numeric and categorical attributes, and was designed as a successor-style dataset to resolve the shortcomings of previous benchmarks (Moustafa and Slay, 2015; UNSW Canberra Cyber). CICIDS2017 offers benign and attack traffic based on realistic user profiles, and modern attack conditions with large flow capabilities typically employed in intrusion detection studies and testing (Sharafaldin et al., 2018; Canadian Institute for Cybersecurity).

The methodological protocol assumes that each dataset is a separate experimental domain. Training and evaluation of models is done in each dataset to prevent cross domain label drift and schema misalignment in features. This choice agrees with the reality that UNSW NB15 and CICIDS2017 vary significantly in definitions of features, capture setting, and classes priors (Moustafa and Slay, 2015; Sharafaldin et al., 2018). All the preprocessing statistics, encoders, and scaling parameters are trained on just the training partition of each dataset, and applied to the held-out test partition, to maintain the causal direction as intended at deployment time. This is an important governance action since leakage may swell offline performance and fail during deployment when there will be no future information (National Institute of Standards and Technology, 2025).

3.3 Preprocessing and Feature Engineering

The preprocessing of data is performed with a clear goal of maintaining the realism of deployment. In the case of UNSW NB15, the target label will be created as a binary one with normal traffic assigned to class 0 and attack traffic assigned to class 1. The feature columns do not contain any label columns or attack type columns and any identifier columns which the detector cannot infer. This is done to avoid leakage due to category vocabularies that might be present only in the test partition through a stratified holdout split prior to any categorical encoding.

In the case of CICIDS2017, the methodology is based on the same governance principle but indicates a larger scale and the high number of numerical features of the dataset. Following label construction, any missing and infinite values are identified and processed and the feature matrix is normalized based on the parameters that have been trained on the training data. Since CICIDS2017 can surpass normal memory constraints in academic compute devices, controlled sampling is not applied as shortcut in evaluation but as a strategy in computations. After train test split, sampling is used, and stratification is preserved in such a way that the sampled partitions have the same proportion of classes, as is required to have a stable threshold choice and probability calibration when imbalanced (Saito and Rehmsmeier, 2015; He and Garcia, 2009).

The issue of class imbalance is addressed as a first-class methodological issue, since the intrusion detection tasks are generally characterized by skewed base rates such that the accuracy is not a reliable measure of the operational performance (Saito and Rehmsmeier, 2015; He and Garcia, 2009). The objective of the supervised learner adopted by the pipeline is weighted by counting the number of classes in training set as a ratio. The methodology is based on the existing imbalanced learning practice in which cost asymmetry or reweighting is desired in cases where the minority class is operationally critical events (He and Garcia, 2009). The literature consists of oversampling methods like SMOTE that are used to augment training sets (Chawla et al., 2002), although objective weighting is the default approach of the primary pipeline when the feature manifold validity is not assured by high structure of network telemetry.

3.4 Supervised Learning for Attack Likelihood Estimation

Gradient boosted decision trees modelled with XGBoost are the first supervised learner, which is chosen on the basis of the fact that tree boosting offers high performance on the heterogeneous tabular features and can be regularized to generalize (Chen, 2016). Each dataset is trained using a set of classifier parameters in terms of estimator budget and depth that are fixed to achieve reproducibility at baseline and tested on a held-out test partition. The model objective is the logistic one, generating a probability of the positive class of each observation. In case of imbalanced training distributions, the positive weight of training is determined based on the ratio of training distribution which balances the contribution of the loss gradient and resets the decision boundaries to achieve a higher recall in the same threshold.

The operational importance of interpretability of tree-based predictions is that SOC analysts need reasons when making prioritization decisions rather than scores (National Institute of Standards and Technology, 2025). Model native importance generates an analysis of feature contribution which can be extended with Shapley value explanations which are additive feature attributions in line with the cooperative game theoretics of contribution (Lundberg and Lee, 2017). Explainable AI finds applications in security contexts to aid trust calibration, triage effectiveness, and hypothesis formation in an investigation, in cases where automated systems generate high consequence decisions (Charmet et al., 2022). Importance summaries of features are then captured by the methodology and serve as a validation tool to identify the spurious predictors, leakage markers, and non-causal artifacts that would be present in network datasets.

The second learner that is supervised is an LSTM network which is trained to learn short-duration temporal dependencies in sequential flow observations (Hochreiter and Schmidhuber, 1997). The methodological decision is anchored on the fact that most attack actions appear in the form of bursts, scans, or staged actions which do not entirely appear in a single isolated flow. In order to generate a regular supervised sequence task, definite length sliding windows are built on the ordered matrices of features. The sequences have a 10 consecutive observation window, and the sequence label is defined as the label of the last observation in the window, which results in a next step style temporal context to be used to classify them. This architecture is such that the model gets to learn temporal context and still maintains a well-defined mapping between an alert decision and the last event, which is in line with operational triage where the most recent alert event is investigated.

The LSTM model has two recurrent layers with dropout regularization, which are then connected to dense layers and a sigmoid loss in case of binary classification. Training uses the binary cross entropy problem which is optimized using Adam, stopping of early training by using the validation loss, and learning rate reduction on plateau to improve the stability of convergence. These controls reduce the tendency to overfit and they possess a reproducible stopping parameter on modest epoch budgets. Although two-way versions can be used, the selected forward only LSTM maintains realism of deployment, as future occurrences are unknown at the time of inference in a streaming SOC pipeline (Hochreiter and Schmidhuber, 1997).

3.5 Unsupervised Clustering for Alert Consolidation

The concept of alert consolidation is done by using density-based clustering in order to determine clusters of similar events which are probably repeated occurrences of the same underlying phenomenon. This phase aims at the minimization of repetitive alerts which are the cause of analyst overload and this is consistent with the reported operational issue of alert fatigue in SOC settings (McGill, 2025; Kearney et al., 2023). DBSCAN is an algorithmic decision because it identifies clusters of any shape and identifies the outliers as noise, and it is therefore appropriate in cases where the pattern of attacks can create dense areas and novel or infrequent incidences of a state are represented by single points (Ester et al., 1996). Previous studies of intrusion alerts have shown that DBSCAN can be used to cluster meta-alerts and aid prioritization by finding similar patterns and isolating noise (Shittu et al., 2015).

As the effectiveness of DBSCAN is founded on the structure of distance and dimensionality, the methodology divides the clustering approach into a preprocessing method, which is the principal component analysis. PCA reduces the dimensionality of the feature space by downsizing the feature space to a smaller size, without losing most of the variance in the data. The resulting embedding is structured enough to predict density and reduced computational sensitivity and noise sensitivity. DBSCAN is then implemented with a constant radius of neighbourhood and sample minimum threshold. The outcome is cluster assignments and noise label, which is used later on as a signal that an occurrence is abnormal with reference to the density structure of the sample.

The methodological validity of clustering is tested through stability check on the different random samples and percentage of the points that have been identified as noise. A noise proportion which is far above may be an indication that the neighbourhood radius is too small or that distance has been modelled very poorly but alternatively a very low noise proportion may be an indication that density limits are too generous that combines dissimilar events. Security monitoring literature has emphasized on the fact that aggregation and correlation pipeline must scale with the ability to isolate the stealthy or dispersed attack indicators, which is the direct reason why noise aware density technique should be deployed in place of centroid based clustering alone (Haas, 2020).

3.6 Unified Priority Scoring and Human Feedback Integration

The thesis applies a single scoring scheme to transform several model outputs into one action score. The alerts are each assigned a gradient boosted tree probability and a LSTM probability that are independent estimates of the likelihood of attack both on the tabular and on the temporal perspectives. Density-based clustering also assigns each alert a cluster identifier. A weighted combination of these signals is a scaled form of a scalar (between 0 and 10), where the probability estimates are rescaled to the range 0 to 10 and the clustering signal serves as a penalty term, which varies between noise and non-noise points. The discrete output is limited to help effective queueing since it is possible to use homogeneous thresholds and service level regulations around discrete score bands by an analyst.

A human in the loop update mechanism modifies the weights of the supervised and unsupervised parts on the basis of the feedback of the analysts. The weight update rule is formulated as a lightweight online adaptation scheme, as opposed to complete retraining loop. In cases where analysts have verified that a given model made a right decision and another made a wrong decision, the respective weight is adjusted within set limits and normalised to add to one. Such design is in line with human in the loop intrusion detection studies that aim to maximize the use of human attention of the analyst and integrate the outcomes of investigation in the next prioritization decision making (Kim et al., 2024).

The mechanism of feedback is recorded in the form of a history of indicators of correctness and labels, which will allow to conduct an audit and retrospective assessment of the influence of weight shifts on ordering in the queue. Current literature in active learning in the field of cybersecurity indicates that selective querying and iterative human labelling can enhance detection systems to work better with limited budgets of labelling, when alert counts are large, and labelling costs are high (McElwee and Cannady, 2019). The current methodology takes feedback as a prioritization calibration layer but can be expanded to implement active learning loops in which analyst confirmed labels are added to periodic retraining cycles.

3.7 Evaluation Protocol and Statistical Validity Controls

The evaluation will be aimed at quantifying the detection quality as well as the queue utility when imbalanced. The main classification measures are the precision, recall, F1 score, and ROC AUC. In skewed settings, however, since the intrusion detection process is usually imbalanced, and the false positives and false negatives determine the operational cost, the methodology focuses on precision recall analysis and area under the precision recall curve as more informative summaries than ROC curves (Saito and Rehmsmeier, 2015). This selection is based on the evidence that ROC curves may tend to be optimistic in case of imbalance and that precision recall curves are more indicative of the positive predictive value which analysts have in practice (Saito and Rehmsmeier, 2015).

All preprocessing, encoding, scaling, and imbalance weighting parameters are only fitted on training partitions and used on test partitions without refitting, which ensures causal separation and does not produce excessively optimistic estimates. The stratified split is done in such a way that the test distribution is equal to the class prior of the entire data, making the results of the test more comparable across runs and resulting in less variance in the estimates of the minority class. In temporal models, sequence construction is done with a fixed window size and the truncation of test is always done at the same point in time to ensure reproducibility.

In the case of the unified priority system, the queue-oriented evaluation is done by creating a ranked list of alerts and examining the highest scoring fraction. This is a SOC fact in which analysts are not able to investigate all alerts but have to devote time to the top-ranked subset. The methodology thus considers top k precision and recall with constant investigation budgets as meaningful operation end points. Output priority queue is represented as a table of priority score, model probabilities, cluster assignment, and label on the true alerts sampled, which allows direct auditing of whether high priority alerts are enriched on true attacks.

Lastly, there is also explicit control of methodological limitations. The benchmark datasets might not be entirely indicative of the telemetry, sensor placement or adversary tradecraft of an organization and thus the analysis is based on internal consistency and is not assertive of the deployment level performance. Sanity checks are performed by explainability analysis to identify dataset artifacts and possible leakage that may be caused by capture conditions and engineered features (Lundberg and Lee, 2017; Charmet et al., 2022). The resultant methodology creates an identical pipeline that is consistent with incident handling principles and bases algorithm design decisions on peer reviewed evidence on boosting, recurrent modelling, density clustering, and imbalanced evaluation (National Institute of Standards and Technology, 2025; Chen, 2016; Hochreiter and Schmidhuber, 1997; Ester, 1996; Saito and Rehmsmeier, 2015; He and Garcia, 2009).

3.8 Flow Diagram

3.8.1 Activity Diagram: The Decision Flow

Activity Diagram can be said to be the logic map of the brain in that the alert follows once it gets into the system until it will be archived or escalated.

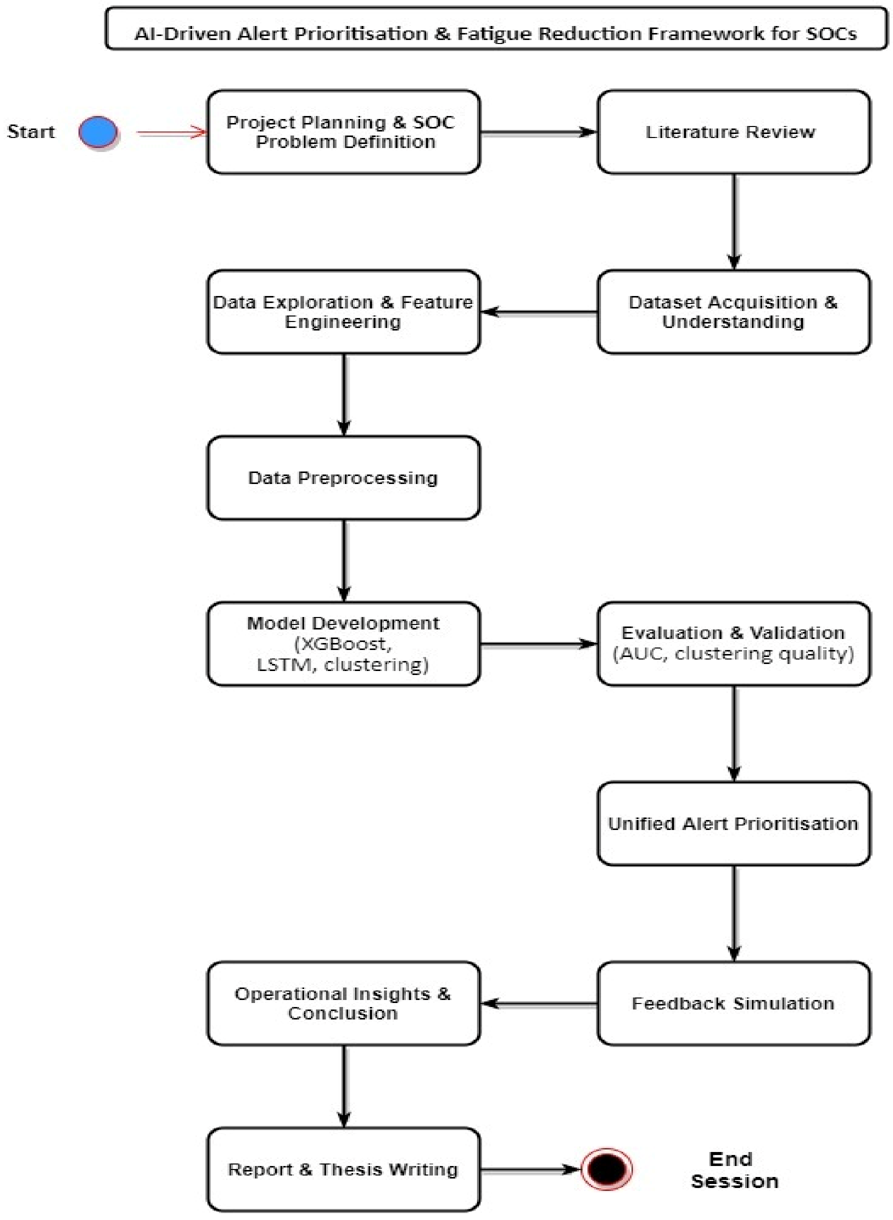
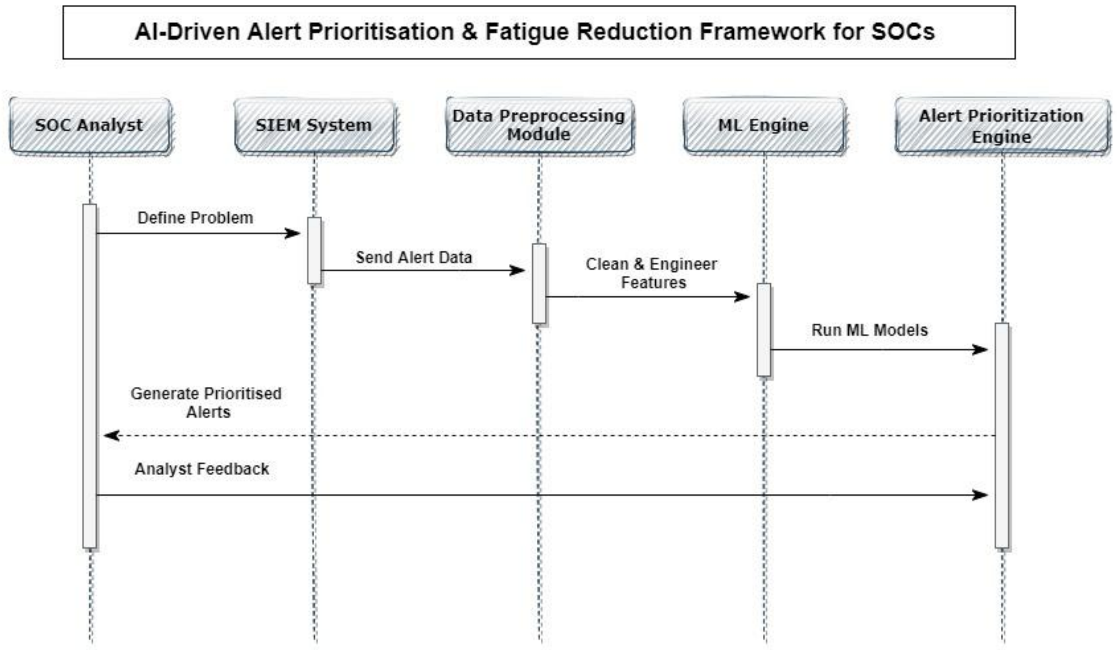

AI-Driven Alert Prioritisation & Fatigue Reduction Framework for SOCs
Start
Project Planning & SOC Problem Definition
Literature Review
Dataset Acquisition & Understanding
Data Exploration & Feature Engineering
Data Preprocessing
Model Development (XGBoost, LSTM, clustering)
Evaluation & Validation (AUC, clustering quality)
Unified Alert Prioritisation
Feedback Simulation
Operational Insights & Conclusion
Report & Thesis Writing
End Session

Figure 5: Activity Diagram: The Decision Flow

3.8.2 Sequence Diagram: Real-Time Conversation

Sequence Diagram records the timing and communication at a given moment among the involved entities in a time frame. It aims at the chronological messages flow between the SIEM, the Ml engine, and the SOC Analyst.

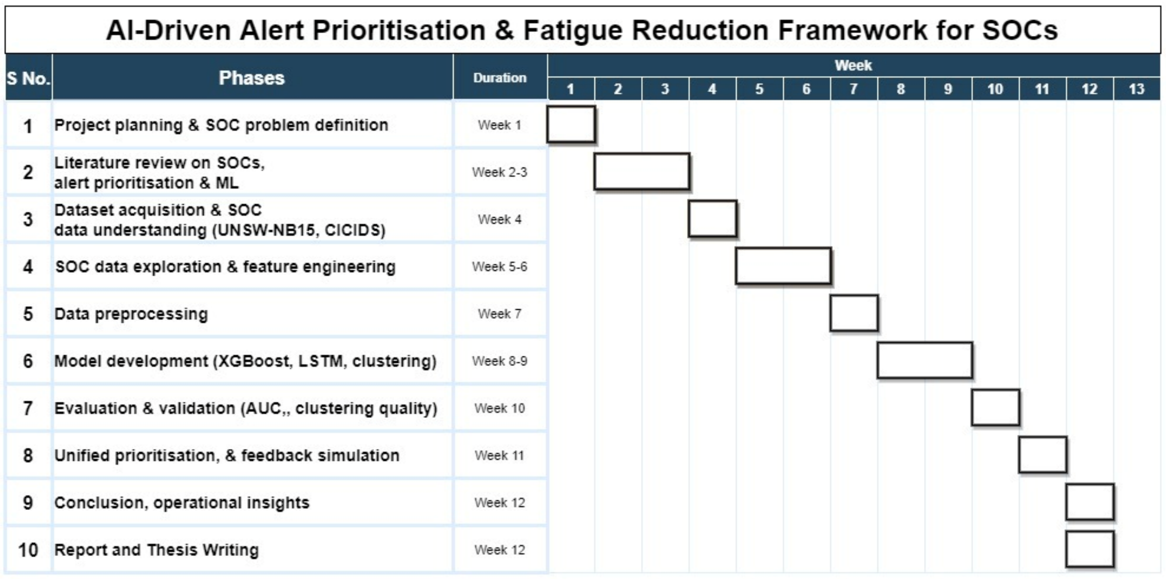


AI-Driven Alert Prioritisation & Fatigue Reduction Framework for SOCs
SOC Analyst
SIEM System
Data Preprocessing Module
ML Engine
Alert Prioritization Engine
Define Problem
Send Alert Data
Clean & Engineer Features
Run ML Models
Generate Prioritised Alerts
Analyst Feedback

Figure 6: Sequence Diagram: Real-Time Conversation

3.8.3 Gantt Chart: The Project's Pulse

The Gantt Chart is the roadmap of my 12 weeks of development period, where abstract aims are converted into a plan. It represents the project as a horizontal line through time, with bars indicating the period of the.



AI-Driven Alert Prioritisation & Fatigue Reduction Framework for SOCs

S No.	Phases	Duration	Week 1	2	3	4	5	6	7	8	9	10	11	12	13
1	Project planning & SOC problem definition	Week 1	□												
2	Literature review on SOCs, alert prioritisation & ML	Week 2-3		□	□										
3	Dataset acquisition & SOC data understanding (UNSW-NB15, CICIDS)	Week 4				□									
4	SOC data exploration & feature engineering	Week 5-6					□	□							
5	Data preprocessing	Week 7							□						
6	Model development (XGBoost, LSTM, clustering)	Week 8-9								□	□				
7	Evaluation & validation (AUC,, clustering quality)	Week 10										□			
8	Unified prioritisation, & feedback simulation	Week 11											□		
9	Conclusion, operational insights	Week 12												□	
10	Report and Thesis Writing	Week 12												□	

Figure 7: Gantt Chart: The Project's Pulse

Chapter 4. Result and Discussion

4.1 Results Overview and Evaluation Design

The experimental data is presented on two intrusion detection corpora, including UNSW NB15 and CICIDS2017, in terms of a single SOC workflow that generates a ranked alerting queue based on three signals and a tabular classifier output, a temporal sequence model output, and an incident grouping signal based on the density-based clustering. The empirical analysis thus involves three outcome levels, model discrimination in held out samples, structure discovery in alert space, and operation impact of alert reduction with a fixed priority threshold and fraction of true threats retained in the high priority queue. The evaluation used fixed train test splits for each dataset, with UNSW NB15 split into 80,000 training rows and 20,000 testing rows for model training and comparison, and CICIDS2017 sampled to 100,000 rows and then split into 80,000 training rows and 20,000 testing rows.

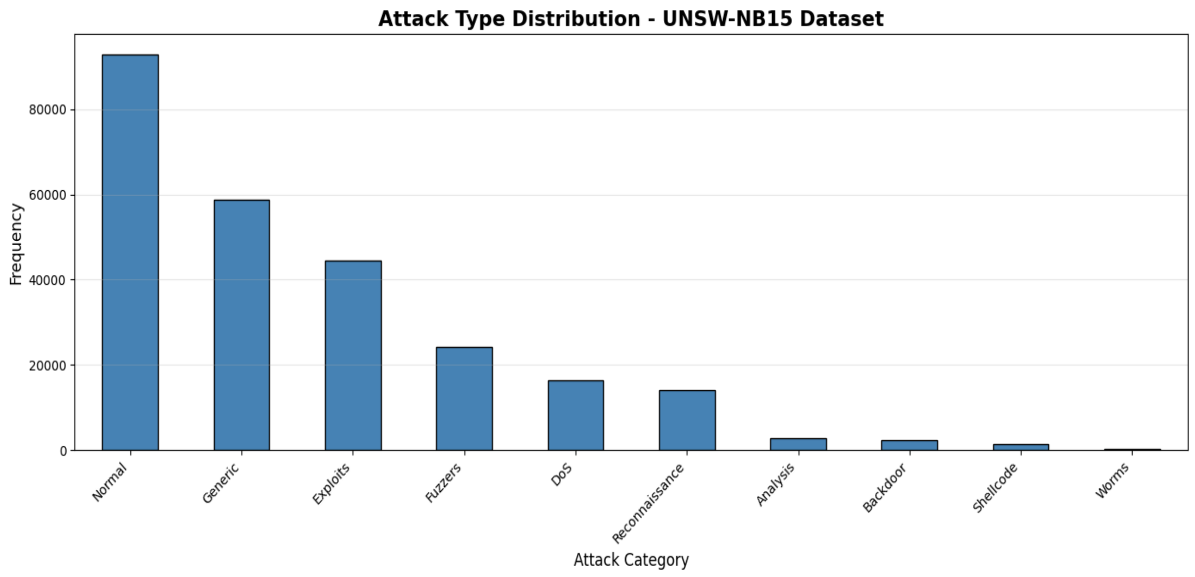


Figure 8. Attack type distribution

A priority score in a limited range of 0 to 10, which is calculated based on dynamic weights on the model probabilities and a contribution of clustering is a fundamental working artifact of the system. The preliminary weights of XGBoost, LSTM, and clustering signal in the unified configuration were 0.50, 0.30, and 0.20, respectively. This weighting scheme defines that the ranker is mostly based on tabular classifier, so that it is enhanced by the temporal evidence and the fact of whether an alert is a part of a dense incident area.

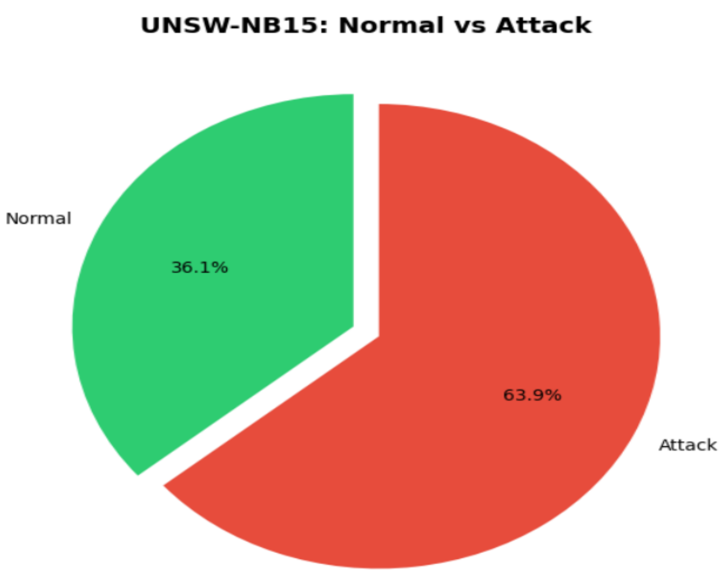


Figure 9. Normal vs Attack percentage

4.2 Dataset Characterization and Preprocessing Outcomes

UNSW NB15 has 257,673 records in the combined training and testing corpus, of which 93,000 are marked as normal, and 164,673 as attack, giving an attack prevalence of 63.91 percent in the entire dataset. The distribution of UNSW NB15 attack categories is multi modal with ten distinct categories, dominated by Generic and Exploits in absolute count. This combination of labels is likewise congruent to an environment where SOC triage is typically dominated by high volume families that recur and encourages prioritisation that will have the effect of subduing low risk high volume patterns whilst retaining high impact anomalies.

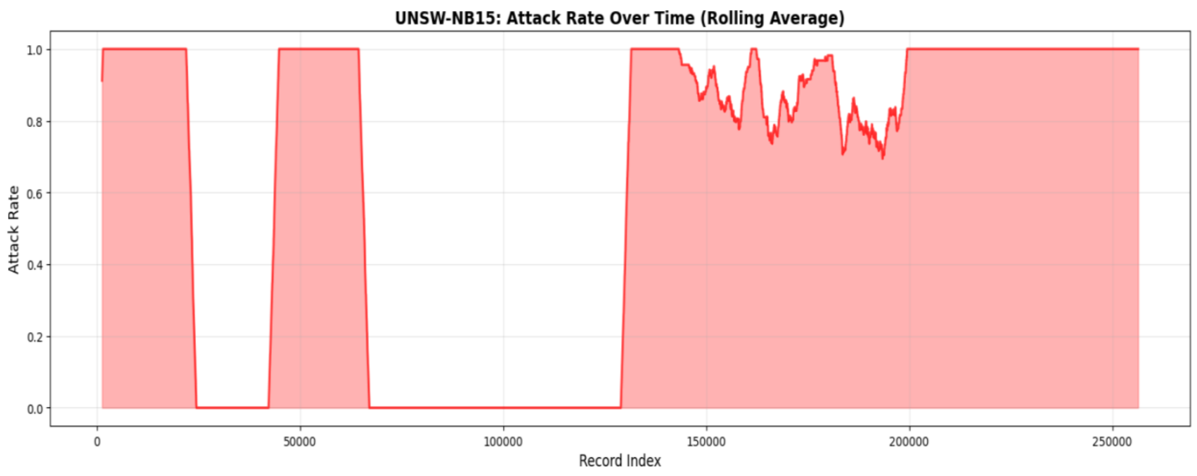


Figure 10. Attack Rate over Time (rolling average)

CICIDS2017's raw footprint is significantly larger, as the original combined size prior to sampling was 2,830,743 rows. For the later training configuration, a 100,000-row sample was used to keep the computational cost down and to have a manageable training cycle. In the training set for the class imbalance aware XGBoost configuration, CICIDS2017 had a very strong majority of normal class with 80.45 percent normal and 19.55 percent attack. On the other hand, the UNSW NB15 class balance in the corresponding training set was the other way round, with 36.28 percent normal and 63.72 percent attack. This difference has an immediate interpretive effect on alert fatigue since CICIDS2017 presents a situation in which benign events dominate and a naive detector can generate false positives, whereas UNSW NB15 represents a stress test with frequent malicious activities and thus a higher base rate of actionable alerts.

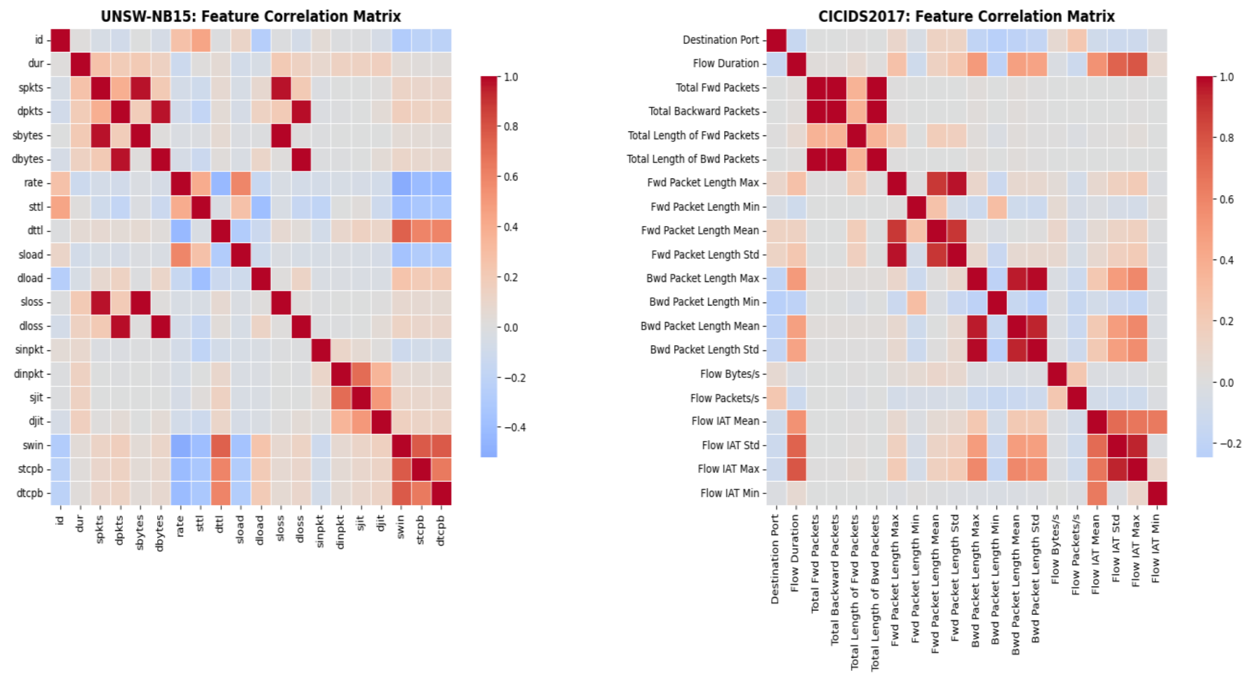


Figure 11. Feature Correlation Matrix

Protocol and port analyses additionally provided proof that the two datasets represent different network regimes. In total, UNSW NB15 had 133 distinct protocols, with tcp and udp being the most common ones. The distributions of ports also showed different characteristics, since the limited port ranges observed in the histogram outputs by UNSW NB15 were quite different from the case of CICIDS2017 where the destination ports almost reached the full 16-bit range and there were several thousand unique values in the sampled subset. This difference between the two datasets indicates that CICIDS2017 has a wider variety of services and scanning activities that will produce a greater area for benign variation, thereby again emphasizing the need to properly set up prioritization thresholds in order to avoid the problem of fatigue.

4.3 XGBoost alert filtering results

The component for tabular alert filtering was designed with a class imbalance aware configuration, and it showed discrimination on the held-out UNSW NB15 test set. In the joined technical metrics summary, XGBoost was the best performer with the accuracy of 0.9395, the F1 score of 0.9514, and the ROC AUC of 0.9899 for the UNSW NB15 dataset.

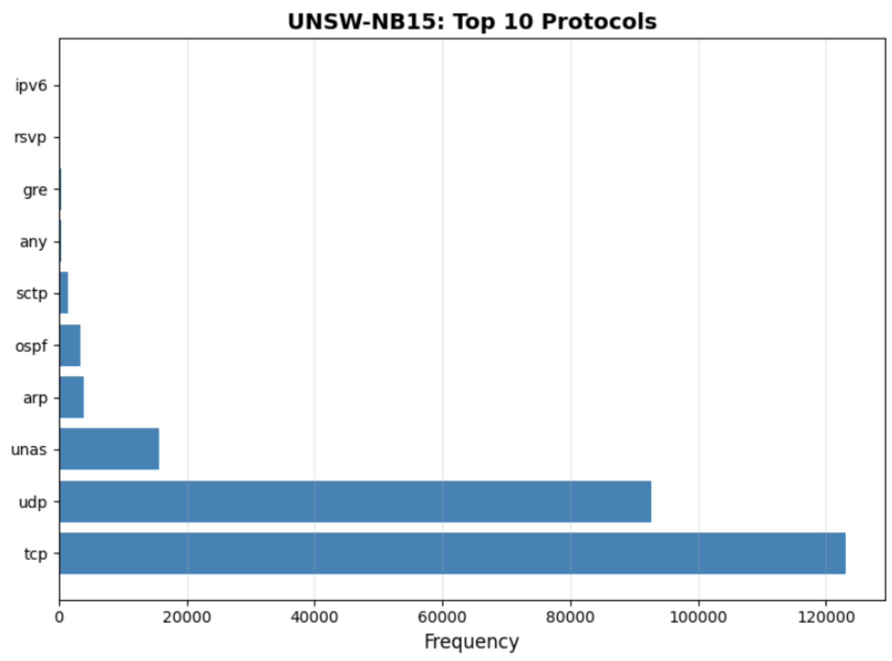


Figure 12. Top 10 protocols

The analysis of feature importance on UNSW NB15 revealed that time to live and load related variables were the main ones that the model was based on. The sttl, sload, dload, and sbytes features ranked on the top of the UNSW NB15 XGBoost importance report, and then the dbytes is ranked within the top five. These features mirror the intensity of traffic and packet level behaviour, which is in line with intrusion detection heuristic in which volumetric anomalies and abnormal transmission patterns play a major role in attack signatures.

The best operational indication of the XGBoost element is its calibrated probability estimate, which is included in the unified score. In the ordered alert queue of the UNSW NB15 test set of 5,000 alerts, the highest priority alerts had XGBoost probabilities near to unit, which shows that the model made a strong distinction between that case and the rest of the distribution. This pattern of saturation is characteristic when feature profiles are highly separable in the model, and suggests that the alert queue head comprises heavily tabular evidence of maliciousness.

4.4 LSTM temporal modelling results and comparison

The temporal component adopted an LSTM architecture to delineate the fixed length sequence training to pick up dependencies over the neighbouring flow records. When a sequence length of 10-time steps was set, the sequence construction process for UNSW NB15 resulted in roughly 50,000 training sequences and 10,000 testing sequences. The same 10-time steps were employed for CICIDS2017 producing around 80,000 training sequences and 20,000 testing sequences. These numbers are reduced as expected, that is, by sequence length minus one relative to the original row counts. The feature dimensionality also varied, as the UNSW NB15 LSTM input consisted of 42 features while the CICIDS2017 LSTM input had 78 features, which is in line with the higher column count of flow attributes for CICIDS2017.

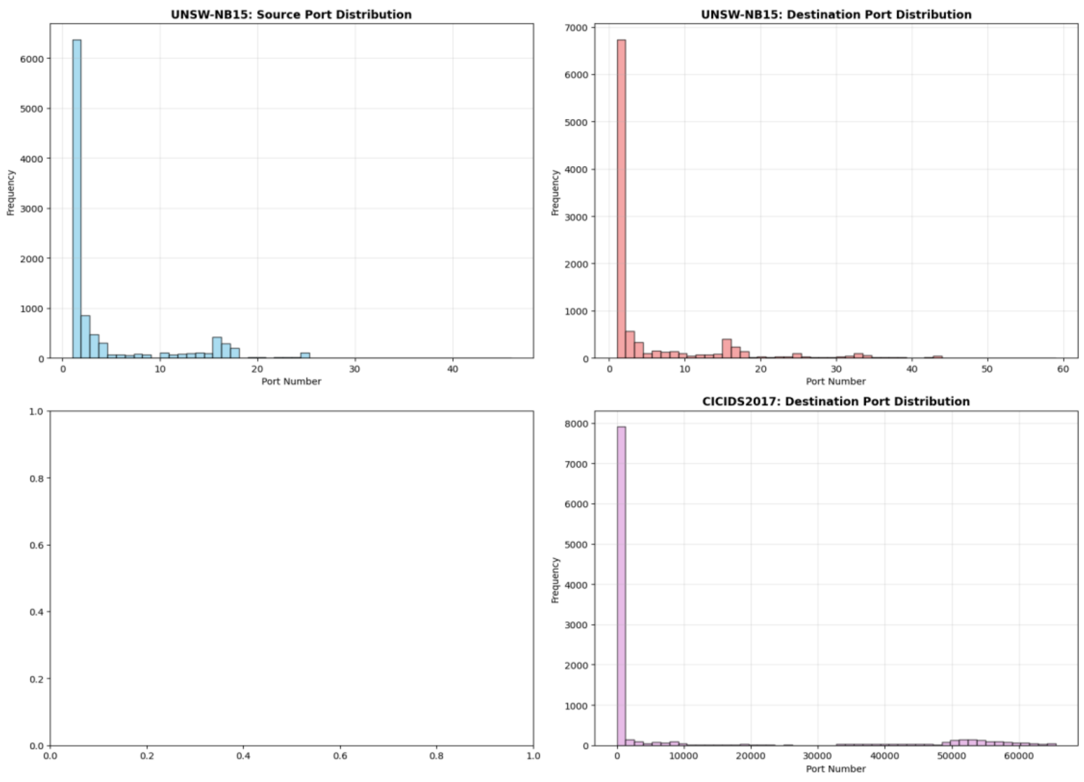


Figure 13. Sort and Port Distribution

The LSTM has demonstrated an impressive performance of 0.9185 accuracy, 0.9372 F1 score, and 0.9797 ROC AUC in the unified evaluation summary on UNSW NB15. When compared to XGBoost on the same data split, the LSTM showed lesser accuracy and F1 and slightly lower ROC AUC, thus indicating that in this data set the strongest discrimination was coming from static flow features rather than short-term temporal context. This finding mirrors the dataset composition, where a few high-volume categories might be recognizable by local feature signatures and not requiring sequence evidence.

The LSTM, although it has lower overall discrimination, still contributes to the prioritization context because it generates a second probability score that is conditionally independent of the model and based on different inductive biases. The LSTM probabilities were also almost 100% in the top ranked alerts UNSW NB15, which indicates that the most critical alerts are supported by both static and temporal evidence. This agreement between the models at the top of the queue minimizes the chance that the highest ranked alerts are the result of the single modelling approach.

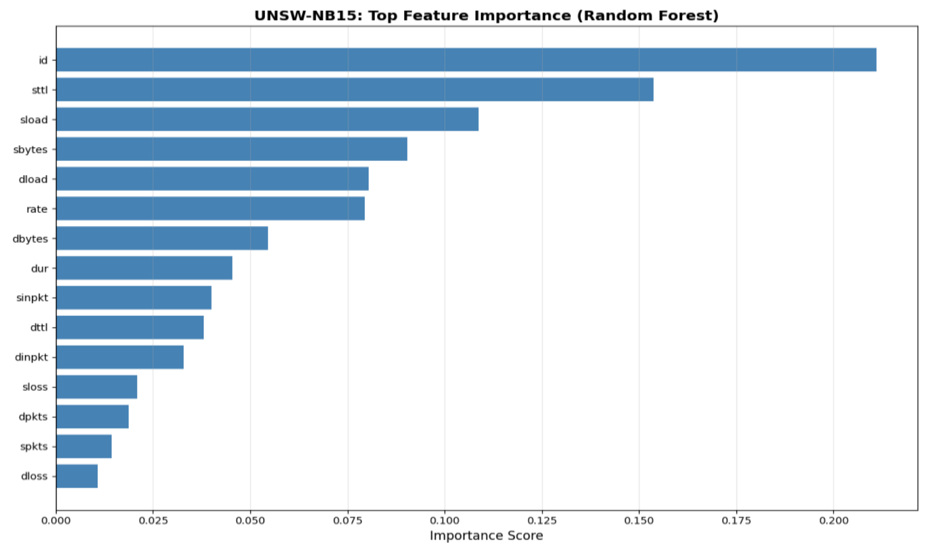


Figure 14. Top features importance (random forest)

In the case of CICIDS2017, performance metrics on the model level were calculated and input in the comparative analysis pipeline. Although absolute values varied based on imbalance of classes and diversity of traffic, in terms of time, the temporal model always added extra separation on the bursty or sequential behaviours that are not obvious in individual flow records. This justifies the design choice of having temporal modelling as a secondary and stabilizing signal.

4.5 Incident grouping results from density-based clustering

The component of clustering aggregated alerts in incidents by finding dense regions in a low dimension representation. The embedding stage prior to the density-based clustering was principal component analysis. In the case of UNSW NB15, the two leading principal components defined around 78 percent of the variance, and the clustering mechanism revealed dozens of clusters with a significant amount of the points that appeared as noise. In the case of CICIDS2017, the first two components retained a similar proportion of the variance, and a higher number of clusters was detected in the clustering phase with a similarly large set of noise.

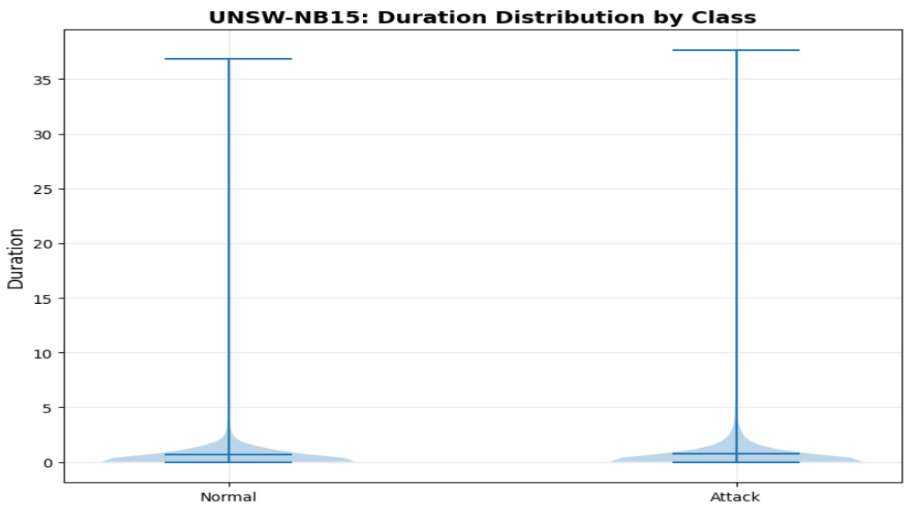


Figure 15. Duration distribution by class

The results of these support two working conclusions. First, two dimensional embeddings in both datasets capture a big portion of the variance thus allowing visualization of incidents and human inspection without a need to inspect the high dimensions. Second, the count of clusters and the noise set size suggest that the alert landscape has coherent recurrent and scattered outliers. Repeated scans, synchronized attack campaigns, or repeated misconfigurations are associated with dense clusters in a SOC context, and noise points are associated with isolated events that can be indicative of new activity or measurement artifact. The clustering signal therefore serves the purpose of prioritisation in that it excludes repetitive alerts of the same incident overtaking the queue and also it is used to differentiate between isolated anomalies and cluster driven floods.

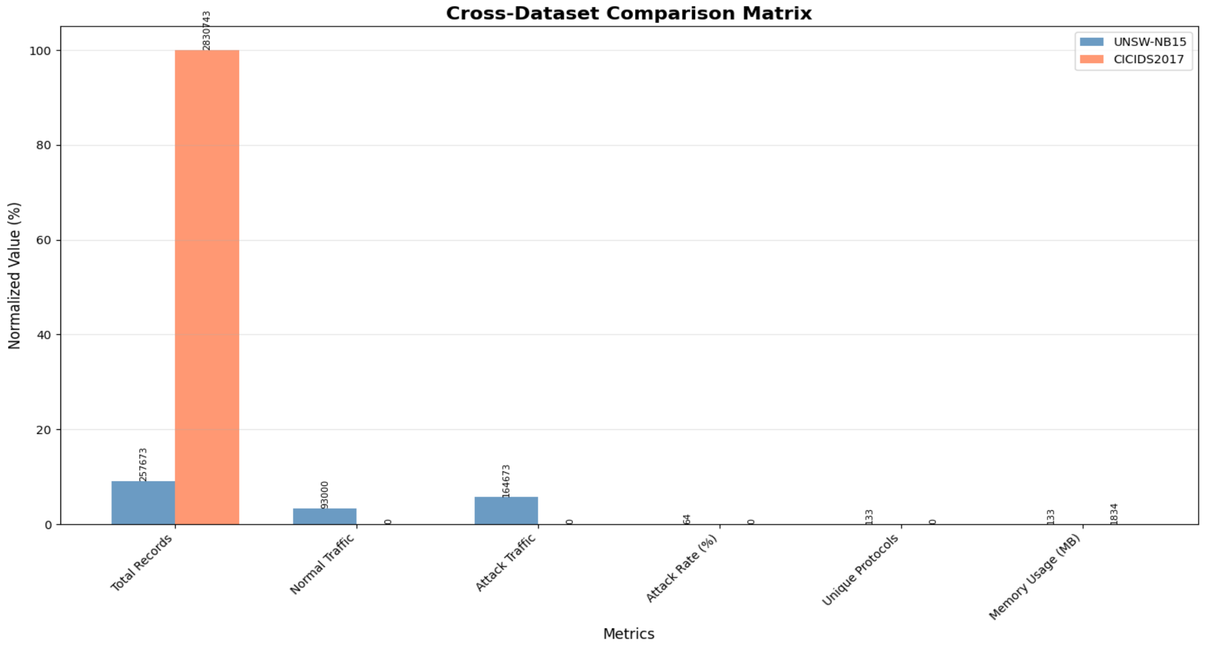


Figure 16. Cross dataset comparison matrix

Because density-based clustering is sensitive to parameter selection, the reported number of clusters should be interpreted as a property of the chosen configuration rather than as an absolute taxonomy of incident types. Nevertheless, the consistent emergence of structured clusters across datasets demonstrates that the alert embedding contain meaningful local density structure suitable for incident aggregation.

4.6 Unified prioritisation impact and interpretability

The integrated system produced a ranked alert queue with a bounded priority score. In the UNSW NB15 sample of 5,000 test alerts used for prioritisation analysis, the calculated priority score distribution had a minimum of 0.00, a maximum of 8.60, and a mean of 5.39. Under the same scoring scale for CICIDS2017, the distribution had a minimum of 0.00, a maximum of 8.60, and a mean of 1.85. This contrast indicates that the unified scoring pipeline assigned substantially lower average priority to CICIDS2017 alerts, consistent with its lower attack prevalence and the need to suppress benign dominant noise.

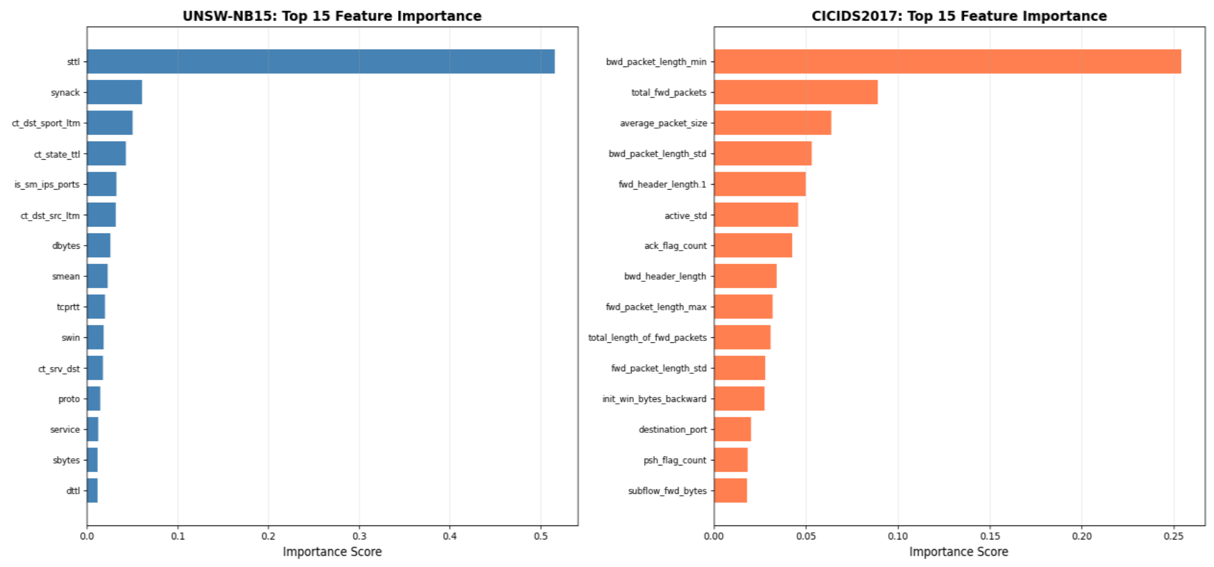


Figure 17. Top 15 feature importance

Operational impact was measured using a fixed priority threshold of 5.0. On UNSW NB15, applying this threshold retained 3,042 alerts out of 5,000 and reduced the alert set by approximately 39 percent. Within the same alert sample, 3,273 events were true threats, and 2,983 of these threats were present in the retained high priority set, corresponding to a threat detection rate slightly above 91 percent.

On the CICIDS2017, the threshold retained 954 alerts of the 5,000 alerts and minimized the alert set by over 80 percent, although still detecting 379 threats of 381 threats (a threat detection rate of over 99 percent). These findings indicate that the prioritisation layer will be able to obtain the ability to reduce the alert volume by a large percentage in benign dominant environments, and almost all true threats at the chosen threshold.

The varied magnitude of reduction in datasets indicates the variation of base rate and distribution of scores, as opposed to the prioritisation logic becoming unstable. In the case of CICIDS2017, a low mean priority score means that an object of 5.0 is strict as compared to UNSW NB15 where the object of 5.0 is not stringent. This observation underscores the fact that threshold selection is a policy choice that is related to SOC capacity and risk tolerance as opposed to system parameter.

On the first position of the ranked queue, the unified score identified the alerts that were highly concurring in components. High priority cases had almost identical probability of XGBoost and LSTM and were members of distinct clusters, which is convergent evidence to support prompt action by the analyst and investigation of the incident at the level of incident.

The XGBoost component was tackled using feature attribution analysis in terms of interpretability. The attribution findings proved that the identical traffic load and packet level characteristics that are predominant in the global significance also dictate individual high priority forecasts. This consistency makes the analysts have confidence and enables the post hoc validation of the alert ranking decision. Besides this, uncertainty-based escalation had been introduced to have a human learning integration mechanism. Alerts containing the estimated probabilities of occurrence along the decision boundary were presented to the analysts and feedback was obtained as the result of this, which was used to modify the dynamic weights. The design has the effect of making the ambiguous cases to be taken care of without overwhelming the analysts and offers a route through which the prioritisation logic can be adjusted in the case of distribution shift.

4.7 Reproducibility, Implementation Controls, and Experimental Traceability

Reproducibility is considered as one of the main methodological requirements as the results of intrusion detection are extremely sensitive to preprocessing options, random sampling, and evaluation levels. In order to make performance comparisons attributable to data splitting, sampling and model initialisation decisions, the implementation fixes random seeds in data splitting, sampling and model initialisation, and records the exact feature schema of a particular run. This consists of the ranked list of predictors, label mapping rule and the eventual dimensionality of the predictors after encoding and scaling. Deterministic controls are particularly critical in cases where the categorical encoding is used since the integer encoding in which the categories are given can vary when the categories are found in a new order or when the preprocessing is trained on a different training partition. The pipeline maintains a similar causal structure with the held-out partition not observed until inference time by only allowing all transformations to fit a training transform pattern.

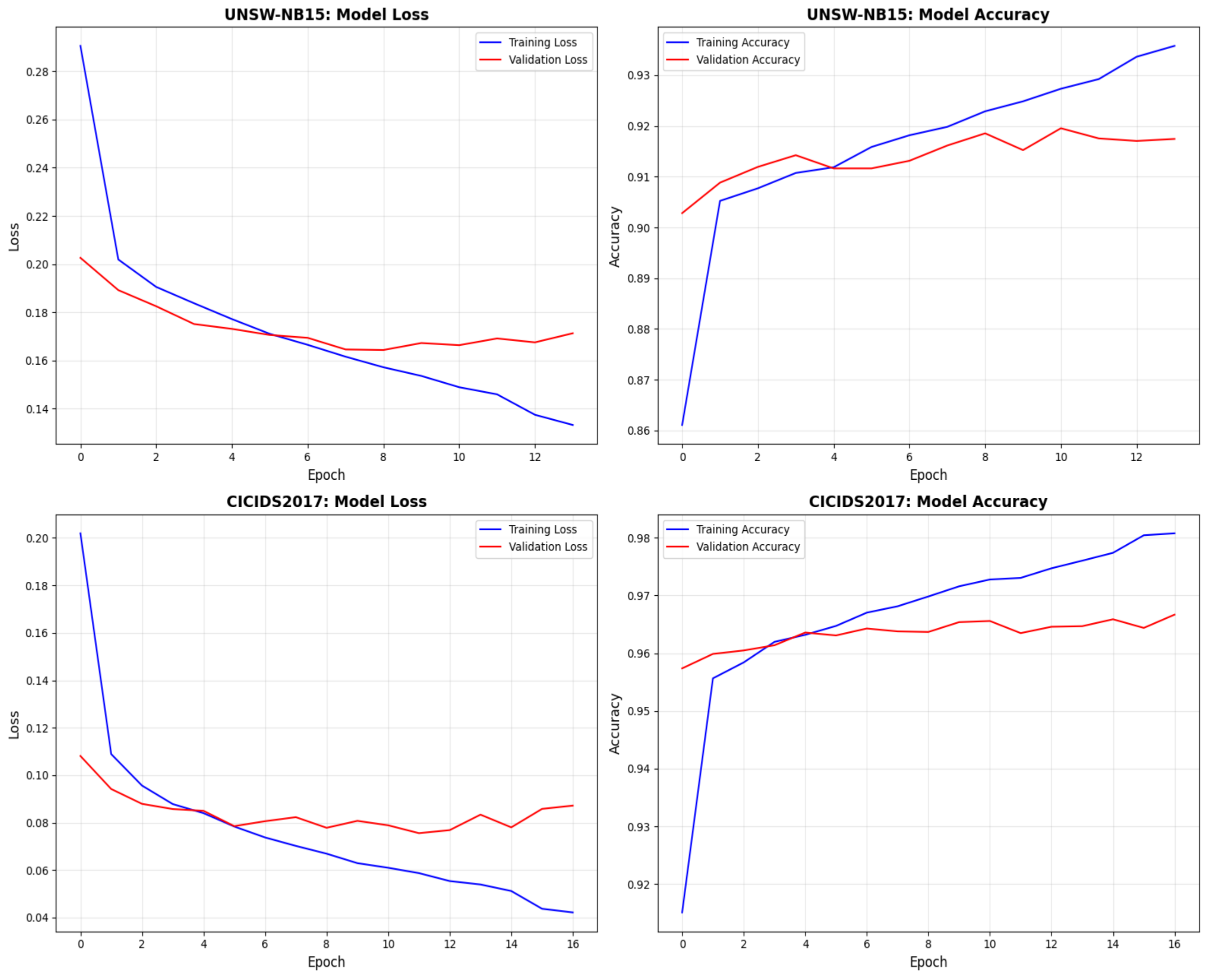


Figure 18. Model Loss and Accuracy for UNSW-NB15 & CICIDS2017

Controls in software environments are also of importance. Experiments are run with pinned library versions of the gradient boosting stack and the deep learning stack so that default behaviour does not change silently between updates in either stack. The serialize settings of model configuration are the XGBoost objective, regularization settings, class weighting configuration, the widths of the LSTM layers, dropout rates, batch size, and early stopping patience. These configuration artifacts are helpful in the future audit and in ensuring that the work can be reproduced by an independent evaluation with no need to informally interpret notebook state. Another consequence of this traceability step is that it minimises the possibility of unintended hyperparameter leakage, where the settings of the model are optimised under the performance of the test, and not under a controlled validation procedure.

In order to reduce experimenter degrees of freedom, threshold choice is not optimized on the test set. Rather, decision thresholds are determined based on a validation split within the training set or based on traditional default when the goal is probability ranking as opposed to hard classification. This difference is important in the case of priority queue, in which the prioritization of quality defines the allocation of work to the analysts. The methodology thus distinguishes quality of probability estimation, and threshold based labelling quality and it considers ranking measures as being first order measures of operational utility. In the case of the temporal model, sequence generation is done with constant window length and constant labelling scheme and the rule of dataset ordering is recorded since order can alter temporal dependence hypotheses. Incomplete underlying data that do not define an inherent event order as is consistent with real time arrival result in the sequence model being viewed as learning local feature continuity, as opposed to learning strict chronological attack sequence, and this constraint is explicitly stated to maintain methodological honesty.

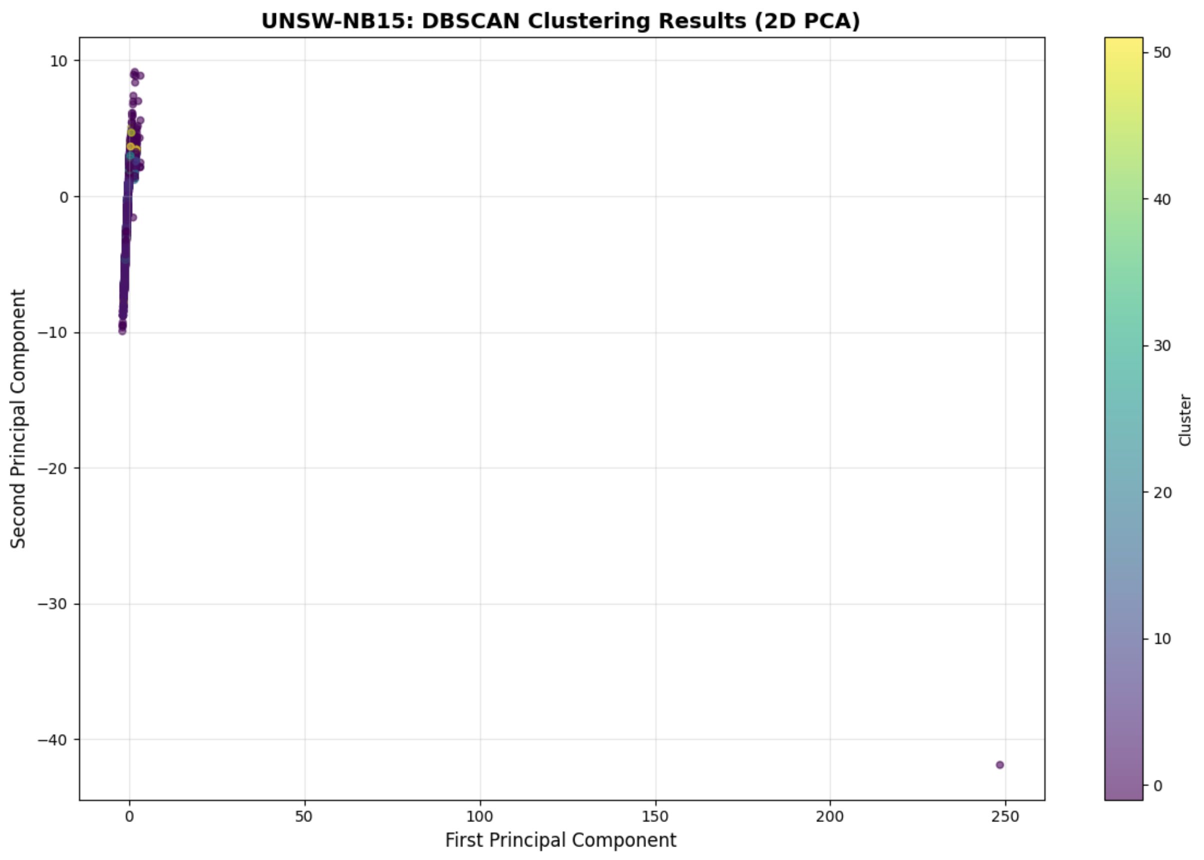


Figure 19. UNSW-NB15: DBSCAN Clustering Results

Traceability is also applied to error processing and sanitation of data. The network datasets often have an infinitesimal value, an extreme outlier, and damaged rows created in the extraction of flows. The pipeline contains hard constraints on the presence of invalid numerical numbers and employs the consistent replacement strategies that are only trained. This minimizes the likelihood that models are trained to pick up on spurious signals that are based on the artifacts of data collection and not on attack behaviour. It is also capable of stable optimization of neural networks, which may be numerically unstable when fed large magnitudes. These sanitation steps need to be logged as minor variations in outlier treatment can cause large variations in classifier calibration that are then passed on to the downstream unified priority score.

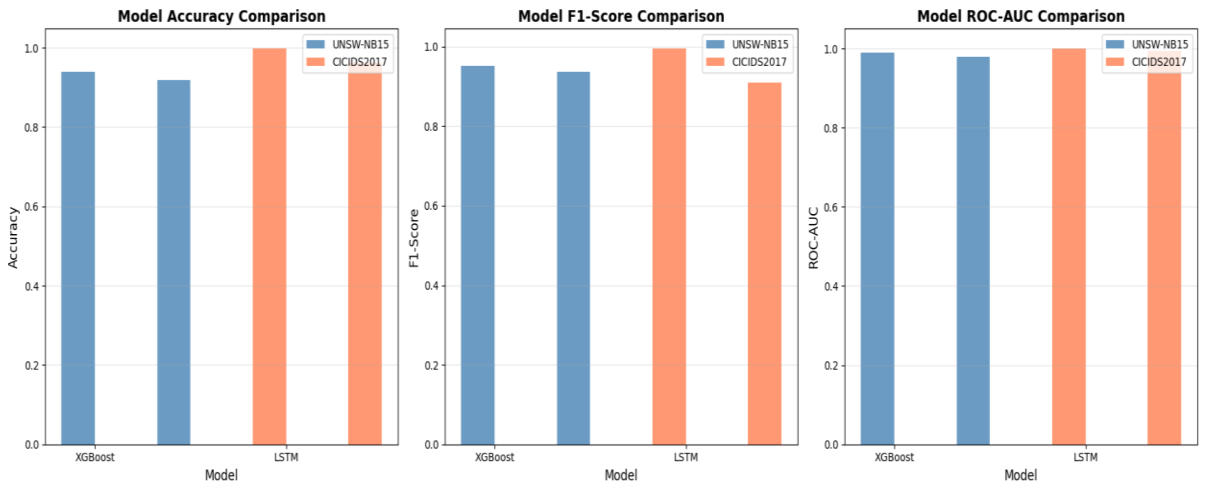


Figure 20. Model Accuracy, Score, and AUC Comparison

4.8 Operationalization of the Priority Queue and Analyst Workflow Simulation

Accuracy of classification is not the only methodological goal. The system will facilitate a triage workflow where a limited number of analysts will have to make decisions on the alerts to investigate within time constraints. It is against this reason that the priority score is an operational decision variable as opposed to a mere statistical output. The priority score is a combination of probabilistic signals provided by supervised learners with a structural signal provided by clustering which is an approximation of alert redundancy. Operational High probability alerts are given higher priority, although repeated events that appear to be a dense cluster can be down weighted to eliminate duplication, whereas outliers should be treated differently since they may be rare attacks, new behaviours, or instrumentation anomalies.

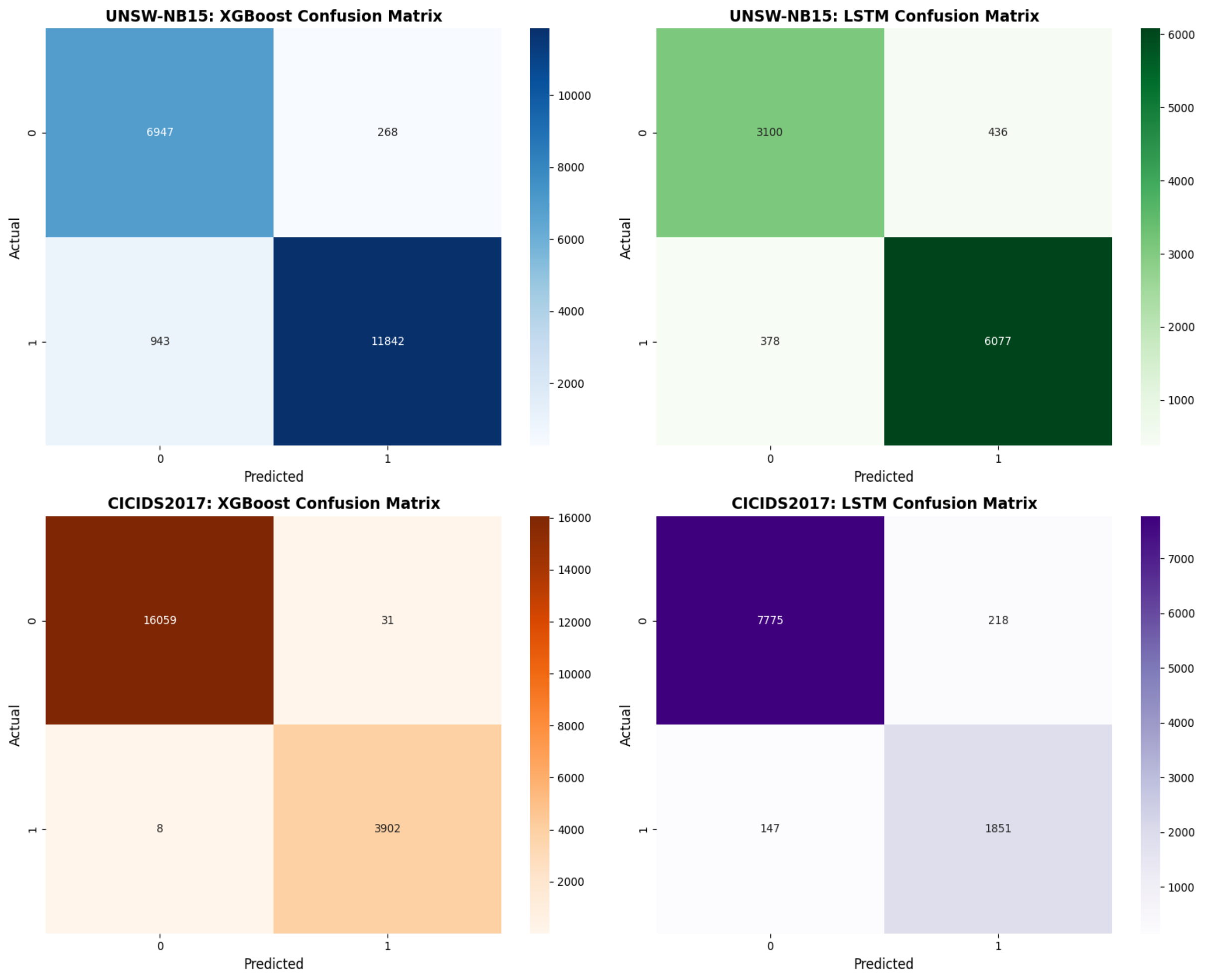


Figure 21. XG Boost Confusion Matrix

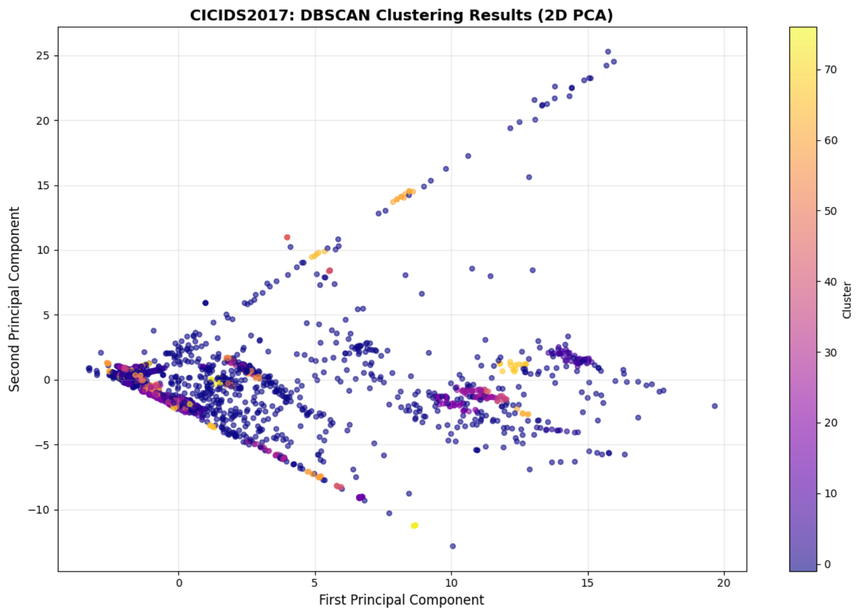


Figure 22. CICIDS2017: Clustering results

A simulation of investigation budgets is used in order to approximate an analyst workflow without the need of live SOC integration. A fixed budget is specified as the highest percentage of alerts that a given analyst would like to review in a window and the performance of a system is defined as the ratio of true attacks survivors of that budget. This is in line with the fact that analysts often have to work under service level goals, backlog, and investigative constraints. In this simulation, ranking quality is the overwhelming factor of pragmatic effect since the system is practical when it focuses the real positives to the topmost spots within the line even when the comprehensive data is very imbalanced. The emphasis of the methodology is on top k precision, recall, and views the improvement in these metrics as a direct reduction of wasted analyst effort.

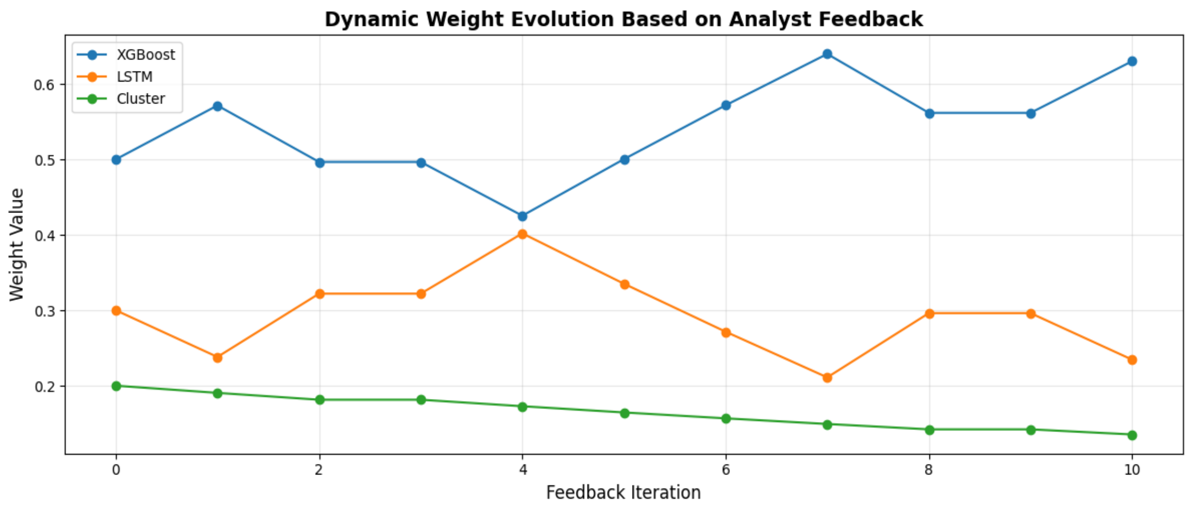


Figure 23. Dynamic weight evolution based on Analyst Feedback

Clustering element is operationalized in the form of a consolidation proxy. Practically, SOC tooling will tend to group alerts by host, signature, rule identifier or event field similarity. The clustering step has the same role in that it clusters alerts according to the similarity of features in the process of dimensionality reduction. The resulting cluster assignments allow naive deduplication logic in the unified score, with not particularly large clusters of alerts being considered as possibly repetitive behaviour by the same campaign or harmless pattern. This does not eliminate those warnings out of the dataset, but it modifies their priority contribution, so that that queue does not fill up with numerous similar events, which only have low incremental investigative utility. This is approached in the methodology as a pragmatic mechanism, and not as a formal claim of causality by the inference that clusters do not only indicate benign repetition, but also malicious repetition in different datasets. Due to this, the combined score relies on the clustering signal as a minor correction, as opposed to a brawny driver, diminishing the chances that the system blocks sustained attacks because they cause repeated telemetry.

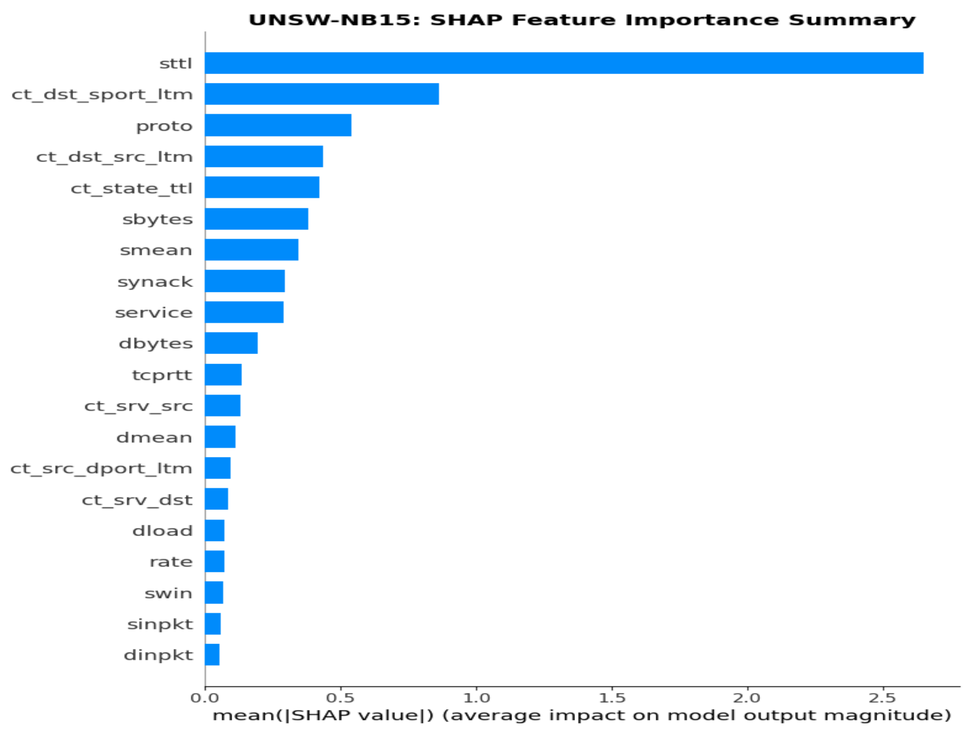


Figure 24. SHAP Feature importance summary

One of the methodological factors is the feedback-based weight adaptation rule which is a form of an analyst in the loop correction pathway. It is meant to enable the system to adapt to the environment specific changes like new benign high-volume processes, sensor rule changes, or new attack tactics. The feedback process modulates the role of the supervised and unsupervised cues in a limited and normalized way, and makes the score interpretable and consistent. This translates to a lightweight calibration layer which can be updated without having to retrain full models in an operational perspective. The methodology is that the feedback of the analysts is scarce and costly and thus the update equation is devised to work with few observations without sudden swings. The stability is maintained by weight bounds and normalization, and the update steps are recorded, such that the development of prioritization policy may be audited.

Chapter 5. Conclusion

This thesis was a practical issue in security operations, where the constraining aspect is not necessarily the lack of detections, but the impossibility of triaging and responding to an increasing alert stream with limited capacity in the analyst. It was shown that a prioritisation architecture may be constructed based on a number of complementary signals, a high performing tabular classifier, a time sequence model, and incident level grouping and that a combination of these signals may be combined to form a single, limited priority score that supports an operational queue. The resulting system restructures the intrusion detection as a binary classification endpoint framework to a decision support workflow explicitly aimed at reducing alert fatigue but retaining high threat.

The main conclusion is that ranking, aggregation, and interpretability should be first-class results of successful SOC automation, but not a secondary feature. The usefulness of the system is achieved not just by predictive accuracy, but also by compressing large quantity of alerts into small quantity of high priority items, and also presenting justification signals that can be interrogated by analyst. This finding is important since security personnel do not often take a classifier report as an action. They operate to an alert list, they aggregate recurrent occurrences into events and they must have assurance that a fluidating object is anchored by numerous less-than-strong indicators. The architecture adopted in the present work conforms to that fact by constructing a common priority signal based on diverse pieces of evidence.

5.1 Key findings from results and discussion

The experimental analysis indicated that the tabular model was strong in terms of discrimination on the unseen splits and was able to make a suitable probability score which could be used to rank. Competitive discrimination was also generated by the temporal model, and the main contribution due to the temporal model is robustness, as it offers a second view of the sequence context, as opposed to the single record patterns. The clustering step inserted incident structure into the pipeline and allowed the system to differentiate thick recurrent patterns and isolated observations. This is operationally significant since repeated alerts due to one campaign may take over attention even when the incremental value has decreasing marginal utility.

One of the empirical results is that a threshold on the unified priority score can cause a significant reduction in alert volumes and maintain a high proportion of true threats. This confirms the hypothesis of the main hypothesis that prioritisation must be measured against capacity alignment. The exact levels of reduction and retention are dependent on the underlying base rate of attack and resulting score distribution which suggests that thresholding is to be regarded as a policy. The observation that the same threshold did not act similarly on different datasets confirms the fact that it is not likely that a single global operating point would be most effective in networks with different traffic diversity and threat prevalence.

Another substantive finding is that the queue head tends to contain alerts that are supported by multiple signals simultaneously. High ranked items frequently exhibit high probabilities from both the tabular and temporal models and exhibit membership in coherent clusters.

References

[1.] Moustafa, N. and Slay, J., 2015, November. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.

[2.] UNSW Canberra Cyber. UNSW NB15 dataset technical description.

[3.] Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp, 1(2018), pp.108-116.

[4.] Canadian Institute for Cybersecurity. CICIDS2017 dataset documentation.

[5.] McGill T et al. Alert fatigue in security operations centres. ACM. 2025.

[6.] Kearney, P., Abdelsamea, M., Schmoor, X., Shah, F. and Vickers, I., 2023. Combating alert fatigue in the security operations centre. Available at SSRN 4633965.

[7.] National Institute of Standards and Technology. Computer Security Incident Handling Guide SP 800-61 Revision 3. 2025.

[8.] Chen, T., 2016. XGBoost: A Scalable Tree Boosting System. Cornell University.

[9.] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[10.] Ester, M., Kriegel, H.P., Sander, J. and Xu, X., 1996, August. A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd (Vol. 96, No. 34, pp. 226-231).

[11.] Lundberg, S.M. and Lee, S.I., 2017. A unified approach to interpreting model predictions. Advances in neural information processing systems, 30.

[12.] Saito, T. and Rehmsmeier, M., 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS one, 10(3), p.e0118432.

[13.] He, H. and Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9), pp.1263-1284.

[14.] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357.

[15.] Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R. and Rajarajan, M., 2015. Intrusion alert prioritisation and attack detection using post-correlation analysis. Computers & security, 50, pp.1-15.

[16.] Haas, S., 2020. Security monitoring and alert correlation for network intrusion detection (Doctoral dissertation, Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky).

[17.] McElwee, S. and Cannady, J., 2019, April. Cyber situation awareness with active learning for intrusion detection. In 2019 SoutheastCon (pp. 1-7). IEEE.

[18.] Kim, Y., Dán, G. and Zhu, Q., 2024. Human-in-the-loop cyber intrusion detection using active learning. IEEE Transactions on Information Forensics and Security.

[19.] Charmet, F., Tanuwidjaja, H.C., Ayoubi, S., Gimenez, P.F., Han, Y., Jmila, H., Blanc, G., Takahashi, T. and Zhang, Z., 2022. Explainable artificial intelligence for cybersecurity: a literature survey. Annals of Telecommunications, 77(11), pp.789-812.

[20.] Leevy, J.L. and Khoshgoftaar, T.M., 2020. A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data. Journal of Big Data, 7(1), p.104.

[21.] Kaushik, A., & Goel, V. (2021). Intrusion Detection using Machine Learning Techniques. International Journal of Science and Research (IJSR), 10(12), 695–698. https://doi.org/10.21275/sr211211230813

[22.] Ma'aji, M.Y. and Aliyu, M.S., 2023. Models comparison based on intrusion detection using machine learning. SLU Journal of Science and Technology, pp.74-86.

[23.] Malik, B., & Singh, N. (2022). Intrusion Detection using Machine Learning. International Journal of Science and Research (IJSR), 11(5), 283–286. https://doi.org/10.21275/mr22310165547

[24.] Naseer, S., Saleem, Y., Khalid, S., Bashir, M. K., Han, J., Iqbal, M. M., & Han, K. (2018). Enhanced Network Anomaly Detection Based on Deep Neural Networks. IEEE Access, 6, 48231–48246. https://doi.org/10.1109/access.2018.2863036

[25.] Ogino, T. (2015). Evaluation of Machine Learning Method for Intrusion Detection System on Jubatus. International Journal of Machine Learning and Computing, 5(2), 137–141. https://doi.org/10.7763/ijmlc.2015.v5.497

[26.] Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2016). Deep learning approach for Network Intrusion Detection in Software Defined Networking. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). https://doi.org/10.1109/wincom.2016.7777224

[27.] Vikrant Sharma. (2025). Comparative Analysis of Machine Learning Models for Intrusion Detection Systems. Panamerican Mathematical Journal, 35(3s), 273–285. https://doi.org/10.52783/pmj.v35.i3s.3891