# *The Design and Implementation of an Intrusion Detection System Using Machine Learning Techniques*

***By***

**Usiwoghene Ekebor**

**Table of Content** **Page Number**


Acknowledgement 5
Abstract 6
1.0Introduction 7
1.1 Background and Context 7
1.2 Problem Statement 9
1.3 Aim and Objectives 9
1.4 Research Questions 10
1.5 Significance 10
1.6 Structure of the Report 11
2.0 Project Scope 12
2.1 Project Overview 12
2.2 Scope of Work 12
2.3 Limitations and Constraints 14
2.4 LESP Considerations 15
2.5 Project Timeline (Gantt Chart Summary) 16
3.0 Literature Review 18
3.1 Intrusion Detection Systems Overview 18
3.2 Traditional vs. ML-based IDS 19
3.3 Dataset Considerations in IDS Research 19
3.4 Gaps in Existing Research 20
4.0 Model Design Methodology 21
4.1 Design Overview 21
4.2 Tools and Technologies Used 22
4.2.1 Python 22
4.2.2 Google Colab 22
4.3 Machine Learning Models Selected 23
4.3.1 Model Justification 24
5.0 Implementation 25
5.1 Development Environment Setup 25
5.2 Dataset Collection and Preprocessing 25
5.2.1 Dataset Description 25
5.3 Data Cleaning and Feature Engineering 28
5.4 Dataset initialization and Model Training 31
6.0 Model Evaluation and Results 32
6.1 Evaluation Methodology 32
6.2 Model Performance Results 33
6.3 Comparison of Models with base research paper 39
6.4 Discussion of Results vs. Objectives 39
6.5 Lessons Learned 40
7.0 Conclusions and Future Work 41
7.1 Summary of Achievements 41
7.2 Research Contributions 41
7.3 Recommendations for Future Improvements 41
8.0 References 43

**Tables and Figures** **Page Number**


Figure 1 Title Cybercrime annual cost in the UK from 2017 to 2028 8

Fig 2. Gantt chart of the project 17

Fig 3 Traditional IDS flowchart 18

Fig 4. Representation of IDS-ML concept 19

Fig 5. Methodology flowchart 21

Fig 6. Flowchart of IDS-ML methodology 22

Fig 7. Flowchart of Decision tree model 23

Fig 8. Representation of ANN 24

Fig 9. Configured Google colab environment for the development 25

Fig 10 Flooding attack 26

Fig. 11 Blackhole attack 26

Fig. 12 Grayhole attack 27

Table 1 Feature explanation of WSN-DS dataset 27

Fig. 13 Correlation matrix of the dataset features 29

Fig. 14 Class distribution of attack type before and after ADASYN 29

Fig. 15 Bar chart of mutual information score 30

Fig. 16 Title the explained variance ratio plot 30

Fig. 17 Data distribution of PCA and LDA 31

Fig. 18 Fitting the processed dataset for model training 31

Table 2 Comparison of trained models 33

Fig. 19 XGB Confusion matrix and class report of initial training 34

Fig. 20 XGB Confusion matrix and class report of PCA applied training 34

Fig. 21 XGB Confusion matrix and class report of LDA applied training 34

Fig. 22 DT Confusion matrix and class report of initial training 35

Fig. 23 DT Confusion matrix and class report of PCA applied training 35

Fig. 24 DT Confusion matrix and class report of LDA applied training 36

Fig. 25 ANN Confusion matrix and class report of initial training 36

Fig. 26 ANN Confusion matrix and class report of PCA applied training 37

Fig. 27 ANN Confusion matrix and class report of LDA applied training 37

Fig. 28 CNN Confusion matrix and class report of initial training 38

Fig.29 CNN Confusion matrix and class report of PCA applied training 38

Fig. 30 CNN Confusion matrix and class report of LDA applied training 38

Fig. 31 Bar chart comparison of the accuracy and detection time of all trained models 39

**Acknowledgement**

Firstly, I thank God Almighty for his strength and wisdom throughout this project. Also, my sincere gratitude goes to my project supervisor, M. S. Mekala, for his support and valuable guidance which made the entire project simple and easy to understand.

Finally, I want to extend my special appreciation to the Security Engineering Lab (SEL) team at Prince Sultan University, Saudi Arabia, for generously sharing the WSN-DS dataset, which played an important role in the successful development and completion of this project.

**Abstract**

This project focuses on developing a machine learning-based Intrusion Detection System (IDS) to improve the detection of cyberattacks in Wireless Sensor Networks (WSNs) and related environments. Recent studies have shown progress in IDS research, yet several limitations exist, many models depend on ideal datasets and fail to perform well in real-world scenarios where traffic is noisy or encrypted. Most Models tend to work on a single dataset but lack generalization across environments such as WSNs. In addition, low-rate Denial-of-Service (LDoS) attacks remain difficult to detect due to their periodic and subtle patterns. Resource-heavy models like CNN-BiLSTM perform well but are unsuitable for lightweight IoT devices. Other challenges include imbalanced datasets, model overfitting, and limited interpretability.

This project leverages on existing studies and applies different machine learning models trained on WSN-DS dataset to compare their performances. Models like Decision Tree, Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and XGBoost were evaluated for accuracy, detection speed and efficiency. To conclude, project presented flexible ML IDS that is more feasible when compared existing models that can detect different DoS attacks with high accuracy and low false positives within seconds.

## 1.0 Introduction

Cybersecurity threats are getting harder to find every day. Cyber attackers now target many types of digital systems, like smart homes, factory networks, and wireless sensor networks (WSNs). Old types of Intrusion Detection Systems (IDS) use fixed rules known signatures to find attacks, but they don't work well anymore (Fares et al. 2025). These old systems have trouble catching new and hidden attacks, especially in places where data traffic changes often, like Time-Sensitive Networks (TSNs) and smart systems using the Internet of Things (IoT).

To fix the problems with old IDS systems, researchers and security experts started using Machine Learning (ML) to make better IDS. ML-based IDS can learn from the data it gets and find new types of attacks without needing updates all the time. But making a good ML-based IDS is not easy. There are problems like having too much of one kind of data, getting too many false alarms, and being too slow to catch attacks in real time (Jihado and Girsang 2024). Particularly, false alarms can overwhelm security analysts and reduce trust in the system, while poor dataset quality can lead to biased or incomplete models.

Moreover, despite these challenges, ML presents as a practical option and tool for improving intrusion detection. Different ML algorithms, such as Decision Trees, Random Forest, XGBoost and others, have shown good results in learning from labelled network traffic data and identifying suspicious behaviour with high accuracy. Nevertheless, no single algorithm performs best in all cases. Trade-offs often exist between detection accuracy, speed, and robustness to noise or imbalance in the dataset being fed to the system (Topsakal, Cevher and Ergenç 2025). Furthermore, intrusion detection in IoT environments requires models that are not only accurate but also lightweight and fast, to support near real-time responses.

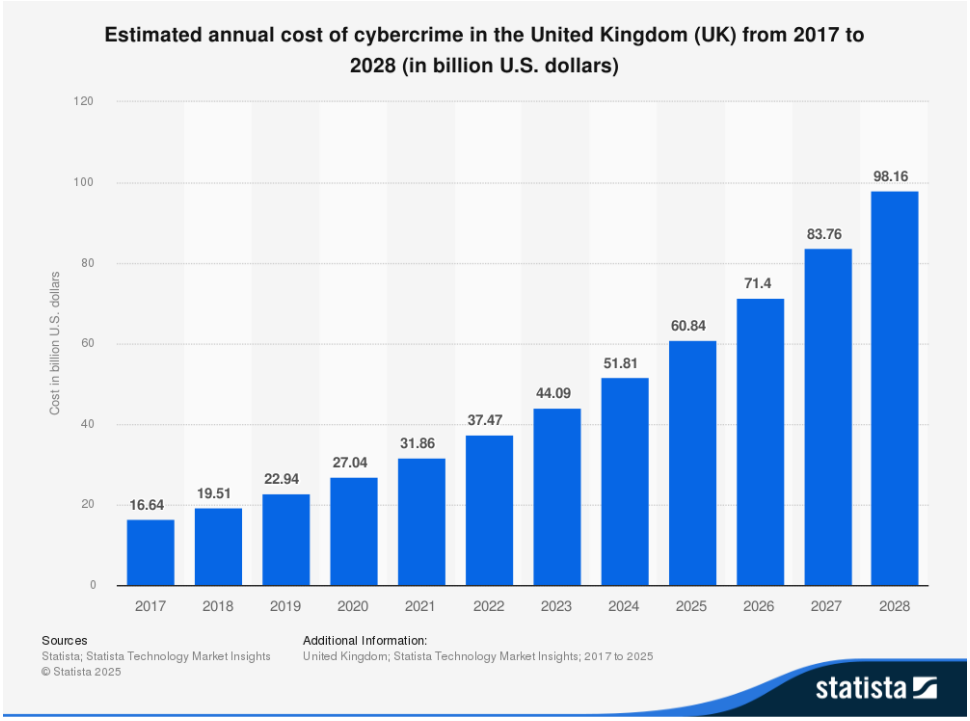
This report presents a project focused on developing and describing the implementation of a machine learning-based IDS designed for modern network environments, particularly smart homes and WSNs. The aim of this project is to reduce false positives, improve detection accuracy, and maintain low latency to meet the demands of real-time systems. Two carefully selected academic datasets will be used to train and test multiple ML algorithms, with a focus being placed on handling imbalanced data and ensuring a powerful performance. Consequently, the Evaluation will be based on metrics such as precision, recall, F1-score, and detection speed.

### 1.1 Background and Context

With the increase of the usage of interconnected digital devices and smart systems, modern networks have become more vulnerable to cyberattacks. The regular security solutions like firewalls and antivirus programs are no longer enough to detect complex cyber threats, mainly in environments like smart homes and Wireless Sensor Networks (WSNs). These systems mostly generate a large amount of data and have unique traffic patterns and signature, which make manual monitoring of intrusion detection both difficult and less ineffective (Fares et al. 2025).

To further illustrate, available data from Statista market insight shows that the estimated cybercrime cost for the year 2028 in the United Kingdom (UK) is projected to go above 90 billion USD, in 2025 it currently sits just above 50 billion USD. The increase in the annual cost explains the gap between cyber-attacks and mitigation strategies like deployment of a ML based IDS. The figure below shows the chart

Figure 1 Title Cybercrime annual cost in the UK from 2017 to 2028




Source Petrosyan (2025)

The aim of an Intrusion Detection Systems (IDS) is to monitor network traffic and alert users and security analyst when suspicious behaviour is detected. While in traditional IDS, the system relies heavily on predefined attack signatures, which made it impossible for this kind of system to adapt to new or evolving attack methods. This has led to the increase of using Machine Learning (ML) to enhance IDS performance, ML trained models can learn from historical data of a network and detect abnormal patterns that Is different from known behaviours, and they also provide options for automation and scalability across various network infrastructures (Fares et al. 2025).

Also, using ML-based IDS brings new problems. The quality and balance of the data used to train the model are very important for how well the model can find attacks. If the data is not balanced, the system may give many false alarms. This can make security workers feel stressed and lose trust in the system. For an ML model to work well in finding intrusions, it must be able to look at live traffic, learn from new data, and give fewer false alarms while still finding real attacks with high accuracy (Topsakal, Cevher and Ergenç 2025).

As a result of these findings, this project uses past research to make a better and flexible IDS using ML methods. It tries to fix the data balance problem and aims to find intrusions in real time with fewer false alarms.

### 1.2 Problem Statement

Cyberattacks are changing quickly, they now target many different weak systems, especially the ones with fewer resources like smart homes devices and wireless sensor networks (WSNs). These smart systems need reliable and fast ways to detect attacks, the detection system must also be able to find both complex and slow attacks with very little delay. Old types of detection systems, called signature-based IDS, often do not work well in these situations. This is because they only look for known attack patterns and cannot easily adjust to new types of attacks. Because of this, some attacks are not noticed. These unnoticed attacks can cause big problems, like stealing of important data, changing how systems work, or stopping services from working.

To solve these problems, many people now use Machine Learning (ML) for detecting attacks. ML seems to be a good method for IDS. But even ML-based systems have problems. One big problem is that the data used to train the systems is often old or does not show how complex today's networks are. This makes the models unfair sometimes or not useful in real life (Al-Ambusaidi et al., 2024). Another major issue is that there are usually more normal data than attack data in the training sets. Because of this, ML models often make mistakes. They may miss rare or hidden attacks, which are called false negatives or they may give too many wrong warnings, called false positives. These false alarms make people lose trust in the IDS (Afroz et al., 2025).

Also, it is still hard to use ML models in real-time systems. Some problems include slow detection, too much use of computer resources, and complicated models. These issues make it hard to use ML in real situations. There is also no single method that works well for all kinds of networks. A good system should balance accuracy, speed, and few false alarms. This shows that more research is needed. This project will try to fix some of these problems by creating and testing an ML-based IDS. The goal is to make a system that is more reliable and works better in today’s network environments.

### 1.3 Aim and Objectives

The main goal of this project is to make and build a Machine Learning system that can find modern cyberattacks in Wireless Sensor Networks (WSNs). The system should be very correct when it finds attacks and should work fast in real-time. At the same time, it must give very few wrong alarms called false positives.

To achieve this goal, the project has some important steps to follow which are.

- **Model Development** – Make a detection model using a labelled dataset. This dataset has both normal traffic and many types of attack traffic. These come from the chosen network places.
- **Handling Imbalanced Data** – Use special ways to choose important features and use oversampling methods. This helps with the problem when there are many more normal records than attack records in the dataset. This will help the model work better with new traffic it has not seen before.

- **Model Evaluation** – Make and test many different Machine Learning algorithms. Check how good they are by looking at their accuracy, how fast they find attacks, and how steady their results are.
- **Testing** – Try the Intrusion Detection System (IDS) in a place without internet, called offline. Use a prepared dataset to see how well it works in real life.
- **False Positive Minimisation** – Make the model better so it gives fewer wrong alarms but still finds many attacks quickly and correctly.

This step-by-step plan makes sure the system is not just good in theory but also works well in real life. Using the right dataset and testing in different ways gives a clear and fair view of how the system performs. By finishing these steps, the project will give useful knowledge about how Machine Learning-based IDS can work in real network settings, especially where the system must be light and respond fast.

### 1.4 Research Questions

This research is guided by four main questions, each addressing a specific challenge in implementing an effective ML-based IDS:

- How well can machine learning detect low-rate and complex intrusion attacks in WSNs?
  This question examines the effectiveness of ML techniques in identifying subtle or irregular attack patterns in different environments.
- What impact does dataset quality and imbalance have on ML-based IDS accuracy?
  This explores how dataset properties, such as class imbalance or outdated attack types, affect the model's learning and detection capabilities.
- Which ML algorithm offers the best balance between detection accuracy and speed for real-time use?
  This investigates which algorithms are suitable for deployment in time-sensitive environments where both speed and accuracy are important.
- How can false positives be minimised while maintaining high detection performance?
  This question looks into techniques for optimising the model to reduce false alarms without sacrificing detection power.

These questions form the foundation for the system design, algorithm selection, testing, and evaluation phases of this project. They also reflect key research gaps in the field of ML-based intrusion detection, as identified in current academic literature.

### 1.5 Significance

As cyber threats keep getting more complicated, especially in Internet of Things (IoT) systems, there is a strong need for smart, flexible, and real-time intrusion detection systems. This project is important because it tries to close the gap between old, unchanging systems and the new smart systems that use Machine Learning. Unlike older methods, this research aims to give fast detection, good accuracy, and easy setup, which are all needed to work well in real-life situations (Fares et al., 2025).

This project looks closely at Denial of Service (DoS) attacks in Wireless Sensor Networks (WSNs). This is important because most older studies on intrusion detection only looked at regular business networks, not these kinds of systems. By testing and comparing how the system works against different attack types, the project helps improve the trust and wide use of Machine Learning-based detection systems.

The project also uses steps like balancing the data, choosing useful features, and creating light models that are easy to run. These steps help the system work better in real-time and lower the number of false alarms, which is a big problem in detection systems. Also, by comparing many Machine Learning methods, this project can give helpful advice to both researchers and security workers on which models work best for different situations.

The results and knowledge from this project will help future cyber defense work, support the making of better detection systems, and help build stronger and safer digital systems for different kinds of connected networks.

### 1.6 Structure of the Report

This report is divided into seven main parts. Each part is made to show the work in a clear and simple order:

- **Introduction** – Outlines the background, research questions, aims, and objectives.
- **2.0 Project Scope** - Defines the boundaries of the work, including limitations, ethical considerations, and a project timeline.
- **3.0 Literature Review** – This part looks at earlier studies about IDS, compares different machine learning methods, and finds what still needs to be studied.
- **4.0 Model Design Methodology** – This part explains how the IDS is planned. It talks about the system's design, the data used, the algorithms chosen, and how the system will be checked.
- **5.0 Implementation** – This part gives details about building the IDS. It includes setting up the environment, training the model, and putting the parts together.
- **6.0 Model Evaluation and Results** – This part shows how the models were tested. It compares the results using different measures and models. It also looks closely at how well the system works and explains what was learned and what problems there are.
- **7.0 Conclusions and Future Work** – This part sums up the results and suggests ideas to improve the system in the future.

At the end of the report, there are references of all sources of materials used. This parts list all the academic papers and sources used in this project. This way, the report gives a full and clear presentation of the project from the first idea to the final work.

## 2.0 Project Scope

### 2.1 Project Overview

Intrusion Detection Systems (IDS) are useful tools that help watch over computer networks and stop people from getting in without permission or doing bad things. Even though IDS are important, the old or traditional IDS are not working well anymore when they try to stop new and smart cyberattacks. This problem becomes even bigger in fast-changing places like smart homes, Wireless Sensor Networks (WSNs), and Time-Sensitive Networks (TSNs).

The main goal of this project is to build an IDS that uses Machine Learning (ML). This smart IDS will be able to find different types of attacks like Denial of Service (DoS) attacks, spoofing (pretending to be someone else), and low-rate DoS attacks in many kinds of networks.

This project will take a close look at some new IDS models to learn what they do well and where they have problems. One good example is a mixed or hybrid model that uses CNN (Convolutional Neural Networks) and BiLSTM (Bidirectional Long Short-Term Memory). This model was made by Jihado and Girsang (2024) and it worked well on new datasets like CICIDS2017 and UNSW-NB15.

The project will also learn from the work of Fares et al. (2025). They studied how to detect many types of attacks in WSNs using a dataset called WSN-DS. Another helpful study is by Topsakal et al. (2025), who worked on finding low-rate DoS attacks in TSNs.

The project will also look at how to detect attacks in smart homes. A study by Afroz et al. (2025) shows how CNNs help in this area. They also talk about future ideas like federated learning (where many devices learn together without sharing data) and blockchain (a secure system to store and share data) to make IDS better in smart homes.

To build this IDS, the project will choose the best parts or features from the datasets. It will also use data cleaning and preparation methods like Principal Component Analysis (PCA). Then, it will train Machine Learning models that can find both attacks we already know about and new attacks we have not seen before.

### 2.2 Scope of Work

This project includes research, planning, building, and checking a Machine Learning-based Intrusion Detection System (IDS). The work will begin by carefully studying different IDS models that already exist. This study will look at both old-style rule-based systems and new types that use Machine Learning. It will also look at systems used in different areas, like Wireless Sensor Networks (WSNs), which were studied by Fares et al. (2025), Time-Sensitive Networks (TSNs), studied by Topsakal et al. (2025), and smart home systems, studied by Afroz et al. (2025).

One important part of the project is choosing and preparing the data that will be used. Popular and trusted datasets like CICIDS2017 and UNSW-NB15 will be used to study general internet traffic. For sensor networks, the WSN-DS dataset will be used. If a small smart

home dataset is available, it will also be used to improve testing for that type of system. The project will also include creating and selecting important data features. It will use methods to reduce the number of features, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). These methods, which are like the ones used by Jihado and Girsang (2024), help make the models work better and faster at finding attacks.

In the next part of the project, the focus will be on training Machine Learning models that learn from labeled data. These include Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), Decision Trees (DT), and Extreme Gradient Boosting (XGB). Each of these models will be tested using important scoring methods like precision, recall, F1-score, and total accuracy. This helps to find out which model is better or worse. To build and test the models, tools like Python, Scikit-learn, TensorFlow/Keras, and Jupyter Notebooks will be used for writing code, training models, and showing results.

At the end, the project will test how well the models work by using special test datasets. This final testing will check how good the models are at finding different types of attacks, especially Denial of Service (DoS) attacks that try to make a network or service stop working. The project will conclude with a reflection on the experimental findings, identifying both achievements and limitations, and providing suggestions for future research directions. All activities will be carried out within a 10-week schedule, supported by regular monitoring and review to ensure steady progress.

### 2.3 Limitations and Constraints

This project is subject to several limitations and constraints that may influence both the implementation and the evaluation of the proposed Intrusion Detection System. The first challenge relates to data availability, access to real-world traffic from smart homes or Wireless Sensor Networks (WSNs) is limited, and most of the widely used datasets, such as CICIDS2017 and WSN-DS, are already pre-processed to a certain level. Even though these datasets are helpful, they do not fully show the newest types of cyber threats or the many different ways real networks work today. Because of this, the models may not work very well when used in real-time situations (Topsakal et al., 2025).

Another problem is with computer power.

Advanced deep learning models, like BiLSTM or mixed CNN-RNN methods explained by Jihado and Girsang (2024), usually need strong computer parts, such as special GPUs, to train faster and work better. This kind of powerful computer equipment is not available for this project because of money limits. This means we cannot use very complex models in this study.

The third problem is the short time available for the project. The duration to complete this project is only 10 weeks, which does not give enough time to explore the research topic in great detail. Because of this short time frame, we also cannot fully test or use the IDS in real-life situations. So, we will only test the models offline, using performance scores, without using real-time learning or feedback from live use.

Another issue is that we cannot use advanced methods like federated learning and blockchain. These methods were suggested by Afroz et al. (2025) as good ways to protect privacy and make IDS systems that are spread out instead of central. But, because we do not have enough time and resources, we cannot include these methods in this project. Still, we will mention them as useful ideas for future research.

Lastly, the models in this project will mostly use supervised learning, which needs datasets that are already labelled. This means the models depend a lot on how correct and clear the labels are in the dataset. If the labels are wrong or confusing, the training will not be good, and the models will not work as well. Even with these problems, the project will still make a working prototype and share useful ideas that can help guide future research in Machine Learning-based IDS.

### 2.4 LESP Considerations

LESP (Legal, Ethical, Social, and Professional) issues must be considered when developing any system to ensure it is safe, fair, and responsible. These principles help protect users' rights, follow laws, avoid harm, and build public trust. Ignoring them can lead to legal issues, bias, or loss of credibility (Finn and Shilton 2023)

**Legal Considerations**

When building this Intrusion Detection System using Machine Learning, the project follows important rules and laws about using data. All the data used in this project, like the WSN-DS dataset and the research papers or information, comes from trusted sources. These are shared for learning and testing only. This makes sure no private or personal details from real people are included, so the project does not break any data privacy laws.

In countries with strong data rules like the General Data Protection Regulation (GDPR) in Europe and the California Consumer Privacy Act (CCPA) in the United States, this project still follows the law. It only uses datasets that are cleaned and made anonymous. These datasets do not have names or other personal details of people. This helps to respect people's rights, such as asking for their permission, keeping data private, and handling it in a safe way.

The project also respects ownership of ideas, called intellectual property (IP). It only uses free and open datasets and tools, like Python, Scikit-learn, TensorFlow/Keras, and Google Colab. It does not use data or software owned by companies or those that require payment. This avoids problems with copyright or limits. Also, no credit card data is used, so rules like the Payment Card Industry Data Security Standard (PCI DSS) do not apply. Still, the project thinks about keeping things safe and controlling access to key parts of the model.

Another legal issue is responsibility when using intrusion detection systems. For example, what if the system fails to see a real attack or wrongly reports normal traffic as an attack? To handle this, the project tests the system carefully and shows results clearly. It checks how accurate the model is, how well it finds attacks, and how often it makes mistakes. It also explains all the steps in building the system, like cleaning data, choosing features, and selecting the right model. This clear way of working is important and follows new AI laws in

the world, like the EU AI Act. These laws say AI must be fair, open, and responsible (Neupane et al., 2022).

**Ethical Considerations**

This project follows important rules about right and wrong when making a Machine Learning-based Intrusion Detection System (ML-IDS). One big rule is to keep user privacy safe. To stop personal data from being shared by mistake, the project only uses data that has no names or personal details. For example, it uses the WSN-DS dataset. These datasets are made for school research and do not have any personal information. This means the data is used in a good and fair way that respects user privacy (Neupane et al., 2022).

The project also thinks about the problem of data bias. If the data is not balanced, the system may treat some users unfairly or miss some bad attacks. To fix this, the project uses ways like adding more data samples (called oversampling), choosing the best features, and making the number of features smaller. These steps help the system be more fair and help it find both common and rare kinds of attacks better (Ntoutsi et al., 2020).

Another big problem is that Machine Learning systems can be hard to understand. To solve this, the project tries to be very clear and open. It tells how the data is prepared, how the models are made, and how the results are checked. The models are tested with easy measures like precision, recall, and false positive rate. This helps other people to understand and trust the system's results (Neupane et al., 2022).

The project also consider the danger of tricky attacks, where someone may try to fool the system. Even though strong protections are not included in this project, the system is tested with different types of attacks to make sure it stays strong and works well in many situations (Vourganas and Michala, 2024).

**Social Consideration**

This project considers social issues when building the Machine Learning Intrusion Detection System (ML-IDS). A common worry is that users might feel like they are always being watched. This can change how people act in places like school, work, or even at home (Hagerty and Rubinov, 2019). But this project does not watch people in real time, so it does not change people's actions or privacy.

It is important that people trust systems like IDS. People may not want to use them if they do not know how the system makes choices. This project helps by making the system open and simple to understand. It shows how the system is checked with accuracy, recall, and false positive rate. It also explains how the models are built and tested. Every step is written down so both researchers and people who care about the work can read and understand (Vourganas and Michala, 2024).

This project also thinks fairness and responsibility are important for people to accept the system. Even if the work does not talk with communities directly, it shows clear results and allows others to check them. This open way helps stop bad effects and supports fairness when using ML-IDS technology.

**Professional Consideration**

In this project, important professional rules were followed very carefully when making and testing the Machine Learning-based Intrusion Detection System (ML-IDS). We used the right knowledge about machine learning, cybersecurity, and how to handle data. This helped to lower mistakes and dangers (Ali et al., 2024).

We also followed professional rules about how to behave. These rules come from groups like IEEE and ACM and they teaches us to be fair, responsible, and to explain things clearly and honestly (Finn and Shilton, 2023). Every part of the project, such as the choices we made in the design, the results from testing, and any changes or updates, was written down carefully. This way, other people can understand and check what we did. We planned to check the project often to make sure the system stays safe, works well, and follows professional standards.

**2.5 Project Timeline (Gantt Chart Summary)**

This project will run for 10 weeks and is divided into important stages, in Weeks 1 to 10, the project will include planning and reviewing research on intrusion detection. In Week 2, the research proposal will be finished, and the required dataset will be collected.
In Weeks 3 and 4, different machine learning models such as ANN, CNN, Decision Tree, and XGBoost will be studied and tested to find the best one for detecting intrusions in Wireless Sensor Networks (WSNs).

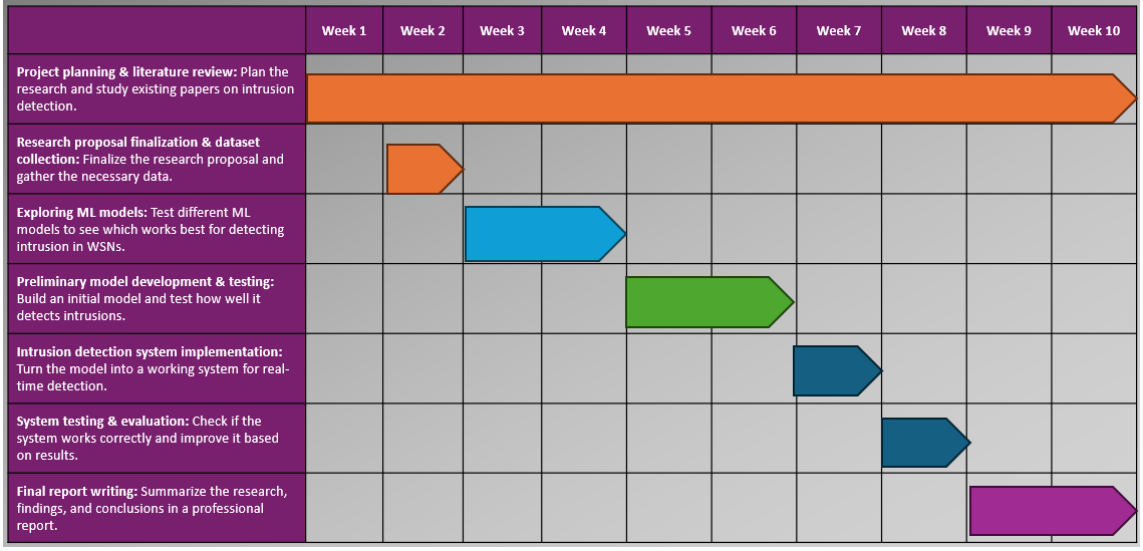
In Weeks 5 and 6, an early version of the detection model will be built and tested to see how well it works, In Week 7, the best model will be turned into a full system for real-time intrusion detection.

In Week 8, the system will go through detailed testing and evaluation using accuracy, precision, and recall. Improvements will be made based on the results. In Weeks 9 and 10, the project will focus on writing the final report and preparing the results for submission and presentation.

A Gantt chart will be used to show the project timeline. It will help track tasks, show links between different activities, and ensure that all important steps are completed on time.

Fig 2. Gantt chart of the project

## Timeline and Project Plan (Gantt Chat)

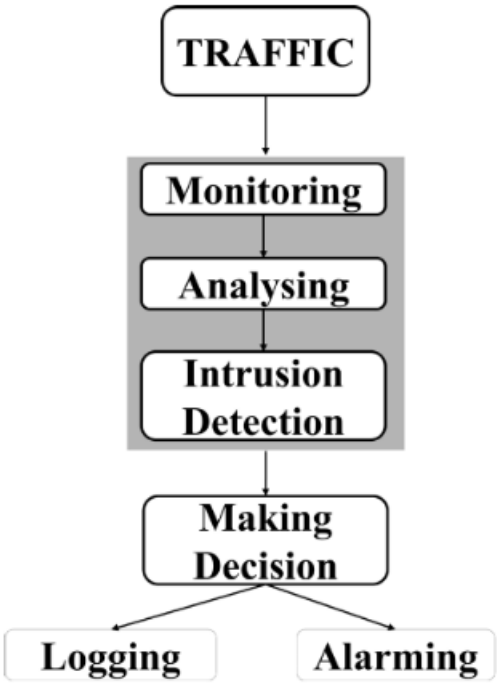


## 3.0 Literature Review

### 3.1 Intrusion Detection Systems Overview

Intrusion Detection Systems (IDS) are tools that help protect computer systems and networks from malicious attacks. There are two main types of IDS; signature-based and anomaly-based IDS. Signature-based IDSs find attacks by comparing data to a list of known attack patterns. Anomaly-based IDSs look for strange behaviour that is different from normal behaviour through the use of statistics or artificial intelligence.

Fig 3 Traditional IDS flowchart




Source Fares et al. (2025)

Today, cyber-attacks are increasing fast and old IDS methods have shown to problems finding new attacks or identifying small changes in known attacks. As a result of this, new IDS systems use machine learning to detect malicious actions in real time. Machine learning IDSs learn from old data and use classification methods to find new or unknown threats efficiently (Fares et al., 2025).

Moreover, in Wireless Sensor Networks (WSNs), IDSs faces problems like low computer power and dynamic network topology. Fares et al. (2025) worked on this problem by using machine learning models trained with the WSN-DS dataset to buld a ML classifier. In Time-Sensitive Networks (TSNs), some attacks like low-rate denial of service are hard to find, leading to Topsakal et al. (2025) to create models that can understand TSN traffic and find these attacks.

In smart homes, where privacy is critical, Afroz et al. (2025) advocate for lightweight, intelligent IDSs that ensure security without compromising user data. These systems can leverage CNNs and future tools like federated learning.

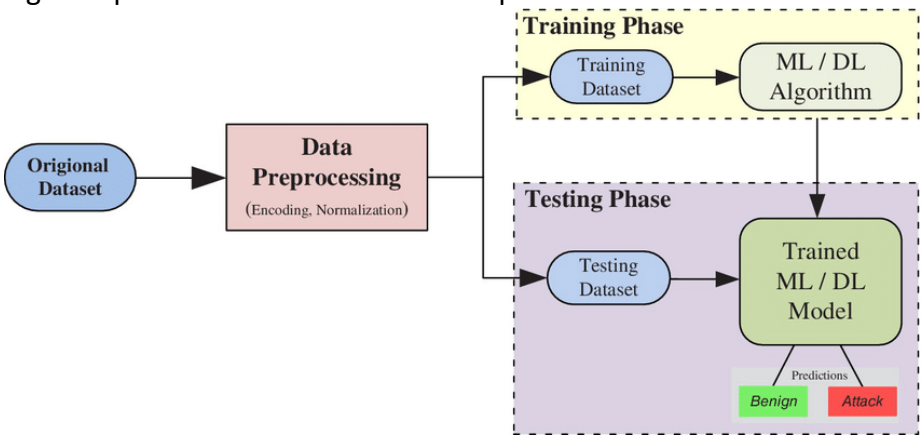
As a result, IDS research is shifting towards adaptable, context-aware systems that incorporate ML to tackle increasingly complex network security challenges.

### 3.2 Traditional vs. ML-based IDS

Traditional Intrusion Detection Systems (IDS) usually use simple rule-based ways to find problems. These include methods like signature detection and anomaly detection. Signature-based IDS look at network data to find known attack signs. Anomaly-based IDS give warnings when the network data looks different from what is normal or usual. These methods work well when the attack is already known. However, they have trouble finding new or unknown attacks. They also make many false alarms because they cannot change easily when new kinds of network data appear (Fares et al., 2025).

Machine learning (ML)-based IDS are better than traditional IDS because they can learn from large amounts of data. They find hidden patterns in the data and can change to detect new attacks. For example, Jihado and Girsang (2024) made a model called CNN-BiLSTM that works better than old IDS. This model can understand both the shape (spatial features) and the time order (temporal features) of the network data. ML-based IDS also work well in difficult systems like smart homes and Wireless Sensor Networks (WSNs), where traditional IDS often do not work well (Afroz et al.).

Fig 4. Representation of IDS-ML concept




Source Ahmad et al. (2021)

ML-based IDS are more flexible and can grow with the system they protect. They can learn using different sets of data, like CICIDS2017 and WSN-DS. This helps them work in many different places and situations. But they have some problems too, they need a lot of computer power to run, they must deal with data that is not balanced, and they need to be updated often to keep working well (Topsakal et al., 2025). Even with these problems, ML-based IDS are seen as more useful and reliable than traditional IDS because they can find unknown threats and reduce how much humans need to help.

### 3.3 Dataset Considerations in IDS Research

The dataset used for ML model training plays an important role in getting accurate and useful results. For this project, different intrusion detection datasets that are freely available to the public were studied. The KDD CUP 99 dataset was checked first, It has been

used in many past studies, but it is too old and does not show recent attack methods, so it is not good for modern intrusion detection. Newer datasets like CIC-IDS2017 and UNSW-NB15 were also reviewed. These datasets include many attack examples, but they do not fully fit the main aim of this project, which is about security in wireless sensor networks. Because of this, the WSN-DS dataset was chosen.

WSN-DS contains special types of DoS attacks such as Blackhole, Grayhole, Flooding, and TDMA. These are common in wireless sensor networks and match the goal of this project. Using WSN-DS helps the models train and test on data that is closer to the real threats in these networks. This makes the results more useful and practical in the target environment.

### 3.4 Gaps in Existing Research

lthough there is progress in IDS research, some problems are still not solved. One problem is that many machine learning models do not get applied in the real world. Most studies train their models on clean and well-labeled datasets, but these dataset do not match real traffic that is noisy and often encrypted (Fares et al., 2025). Many models also work well on one dataset but fail when tested in other environments like smart homes, TSNs, or WSNs (Afroz et al., 2025).

Another problem is weak detection of low-rate DoS attacks. Topsakal et al. (2025) explain that normal methods and many ML models cannot detect LDoS in Time-Sensitive Networks because of their slow and hidden attack pattern. For smart homes, IDS models often forget about privacy and resource limits, so they are not lightweight or secure enough (Afroz et al., 2025).

Some models such as CNN-BiLSTM by Jihado and Girsang (2024) show good results but still need strong hardware, which makes them less useful for IoT devices. Few studies test new ideas like on-device training, which could help with these limitations, additionally, issues like imbalanced datasets, overfitting, and model interpretability remain under-addressed.

## 4.0 Model Design Methodology and Implementation

This project model development followed a structured process starting from data acquisition, preprocessing, 4 model training, and performance evaluation Fig 5.

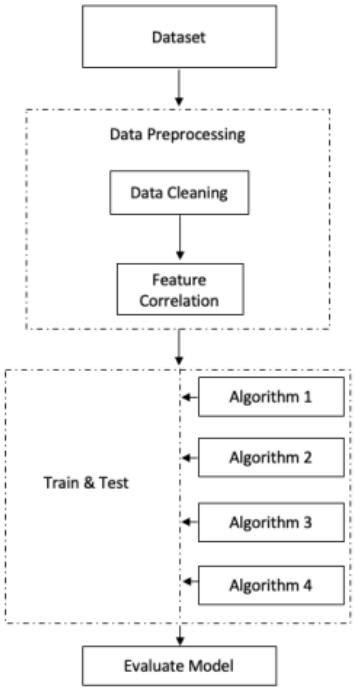


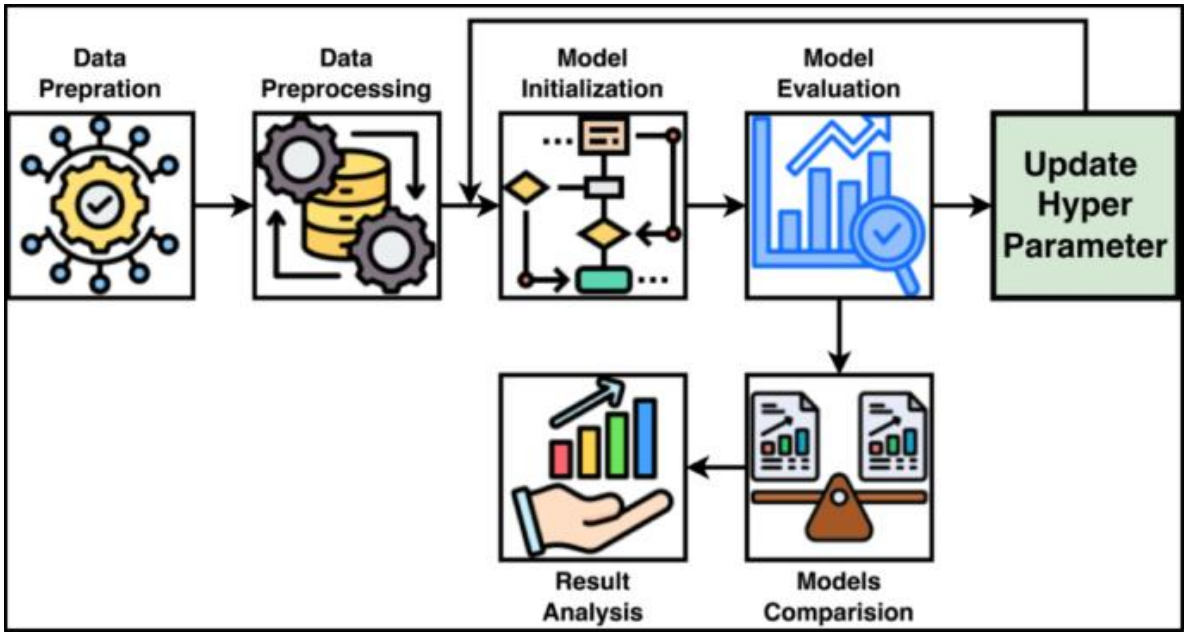

Fig 5. Methodology flowchart

### 4.1 Design Overview

The Models were designed to detect four types of DoS attacks: Blackhole, Grayhole, Flooding, and Time-division multiple access (TDMA), alongside normal traffic. After loading the dataset into the IDE, the dataset was further explored which helped in identifying its structure and imbalance issues. To improve detection performance of the dataset was cleaned by dropping columns that were not important, then sampling of the dataset before model training.

Sampling of the dataset was done using Adaptive Synthetic Sampling (ADASYN), which gives priority to underrepresented classes and increases learning in complex regions. This was followed by model training using various classifiers. After the initial results, a feature engineering step was added to the pipeline, using Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to decrease dimensionality and reduce noise. Moreover, the output of the models was evaluated using the performance metrics to identify the compromise between detection accuracy and time. The design of these models is aimed to create a lightweight and accurate intrusion detection model suitable for real-time systems.

Fig 6. Flowchart of IDS-ML methodology



Source Pandey et al. (2025)

### 4.2 Tools and Technologies Used

Some key tools were utilized during the developing stage of the machine learning-based IDS models of this project. Some of these essential tools are highlighted and briefly explained below.

#### 4.2.1 Python

Python is a general-purpose programming language known for its clear syntax and rich ecosystem. It's popularly used in machine learning, data analysis, web development, and automation due how versatile it is. To use python properly for the model development of this project, some important library was imported some of which are explained below.

- **Pandas -** Pandas is a Python library that provides a high-level data structures and functions for data analysis. Pandas introduces DataFrame and Series objects for handling structured data efficiently which makes it very essential for data preprocessing.
- **Scikit-learn (Sklearn) -** Scikit-learn is a powerful machine learning library in Python which offers simple and efficient tools for data mining and analysis, some of which are classification, regression, and clustering, with a consistent interface for all models.

#### 4.2.2 Google Colab

Google Colab is a free and paid, cloud-based Jupyter notebook environment that supports Python. It allows users to write and execute code in the browser, with free access to CPUs and paid access to GPUs, which makes it ideal environment for machine learning training of this project. For this project, Google Colab Premium was used allow GPU acceleration and seamless integration with Google Drive for file storage and dataset access. This setup enabled efficient experimentation with the training and testing datasets without requiring local system resources.

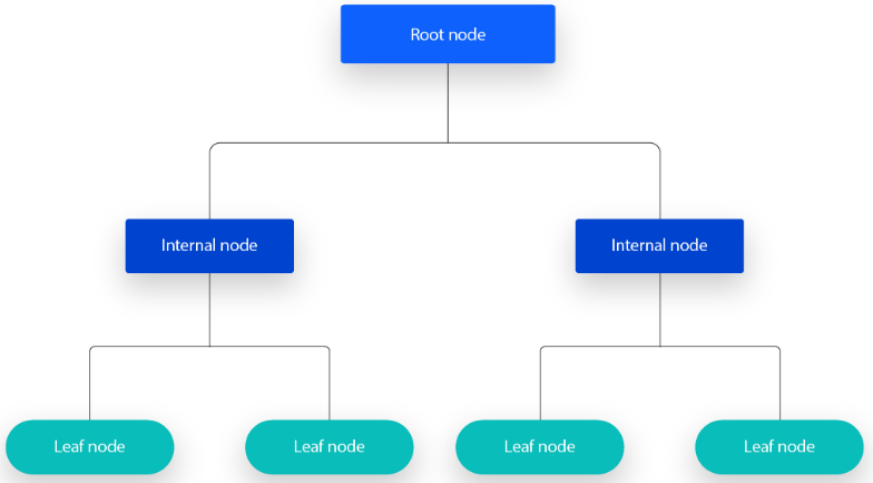
### 4.3 Machine Learning Models Selected

A total of four models were selected and trained with the WSN-SD dataset to evaluate their effectiveness in detecting intrusions: Decision Tree, XGBoost, Convolution Neural Network (CNN) and Artificial Neural Network (ANN). These were chosen to represent both traditional classifiers and advanced ensemble methods.

**Decision Tree**

A Decision Tree is a supervised learning algorithm that splits data into branches to make predictions. It is easy to interpret and suitable for both classification and regression tasks, although it is prone to overfitting if not pruned.

Fig 7. Flowchart of Decision tree model




Source IBM

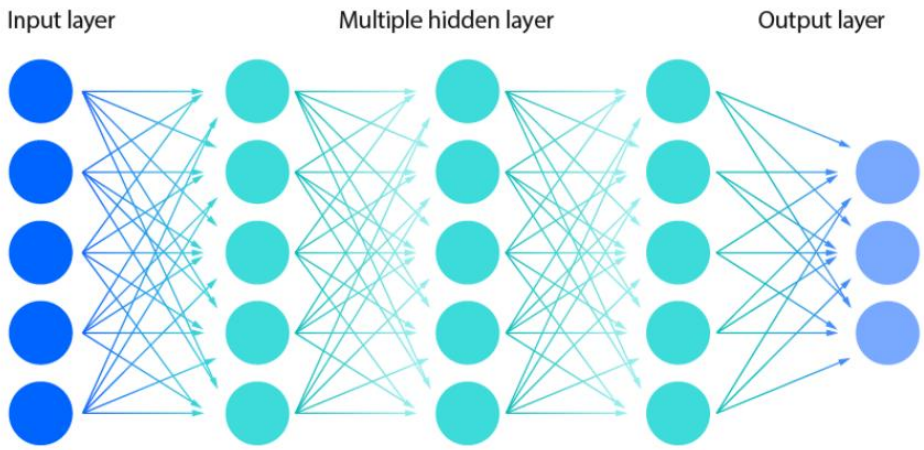
**XGBoost**

XGBoost (Extreme Gradient Boosting) is a high-performance implementation of gradient boosting. It includes regularization to reduce overfitting and handles missing values automatically. It is known for its accuracy and speed.

**Artificial Neural Network (ANN)**

Artificial Neural Networks is computational model inspired by the brain, it consists of interconnected layers of neurons. They learn patterns from data and are used in deep learning for complex tasks.

Fig 8. Representation of ANN




Source IBM

**Convolutional Neural Network (CNN)**

A Convolutional Neural Network (CNN) is a machine learning model that is deep forward and employ pooling and a fully connected layer to extract feature through their hierarchy from raw data which enables efficient end to end learning of the model.

### 4.3.1 Model Justification

The selected Models were chosen based on their strengths in handling classification tasks and computational efficiency; these models are suitable for the nature of the dataset used for this project.

Decision Tree and Random Forest were selected for their ability to handle non-linear relationships and high interpretability.
XGBoost were chosen for their gradient boosting framework, which provides fast training and strong generalization.

ANN was used due to its ability to model complex patterns which are very common WSN behaviours. Moreover, the numerical features of the WSN-DS dataset allow ANN to learn the relational pattern in a network behaviour which could guarantee over 90% accuracy in detecting an attack in WSN (Jingjing et al. 2022).

CNN also proved to be very effective for intrusion detection with the WSN-DS dataset because not only does it handles image data, but it also has the capability of handling high dimensional numeric data like the ones in WSN-DS dataset to achieve high accuracy (Alhasan and Hamza 2023).

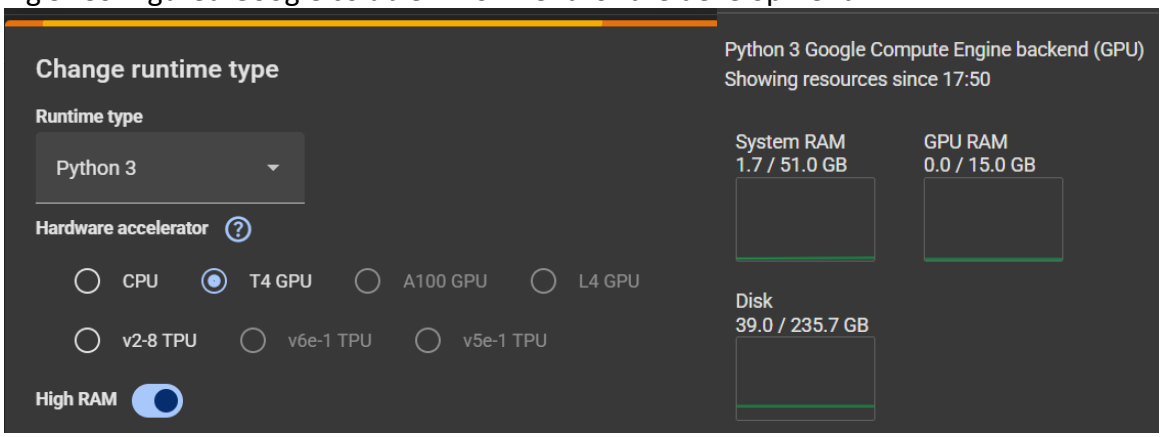
Consequently, this project took advantage of the different strengths of these models and make comparison on which model even with multiple feature engineering would be suitable for a live production environment.

## 5.0 Implementation

### 5.1 Development Environment Setup

The development environment used which is Google Colab premium, was set up with dynamic CPU and GPU support. It provided guaranteed access to 12GB of RAM and 250GB of storage. Although, there is non-guaranteed access to 51GB of RAM, 15GB of GPU memory, and 235GB of storage, as shown in Fig. After setting up the environment, a new Python notebook was created to begin the project.

Fig 9. Configured Google colab environment for the development




After creating the new Python notebook, the necessary libraries for the project were imported. Then, Google Drive, where the dataset was stored, was mounted, and the dataset was loaded into the development environment.

### 5.2 Dataset Collection and Preprocessing

The dataset used for the model training was collected through formal request and shared by the Security Engineering Lab (SEL) of Prince Sultan University. The dataset contains 374,661 records and 19 features, stored in CSV format. The set was loaded it into Google Colab, the structure and contents were explored to understand the distribution of class labels.

A major issue identified was the strong imbalance between normal and attack records. To correct this, ADASYN was used to generate synthetic samples for minority attack types. Unlike Synthetic Minority Oversampling Technique (SMOTE), ADASYN focuses on complex data regions where classifiers are likely to misclassify, making it more suitable considering the dataset imbalance and overlap.

Feature selection was then applied to remove less useful attributes, reducing features from 18 to 13. This was followed by a 70/30 train-test split. To improve model output, PCA was applied to further reduce the features from 13 to 9 features, which enhances training speed and reduces overfitting. Subsequently, LDA was also applied to get a better performance from the trained models.

#### 5.2.1 Dataset Description

The dataset used in this project consist of 19 features and a total of 374,661 records. These are divided into five classes: Normal (340,066), Grayhole (14,596), Blackhole (10,049), TDMA (6,638), and Flooding (3,312).

**Flooding**: This is a kind of DoS (Denial of Service) attack. It happens when a hacker sends a lot of traffic to a system. This makes the system busy and stops it from working properly. Figure 10 shows how this attack looks.

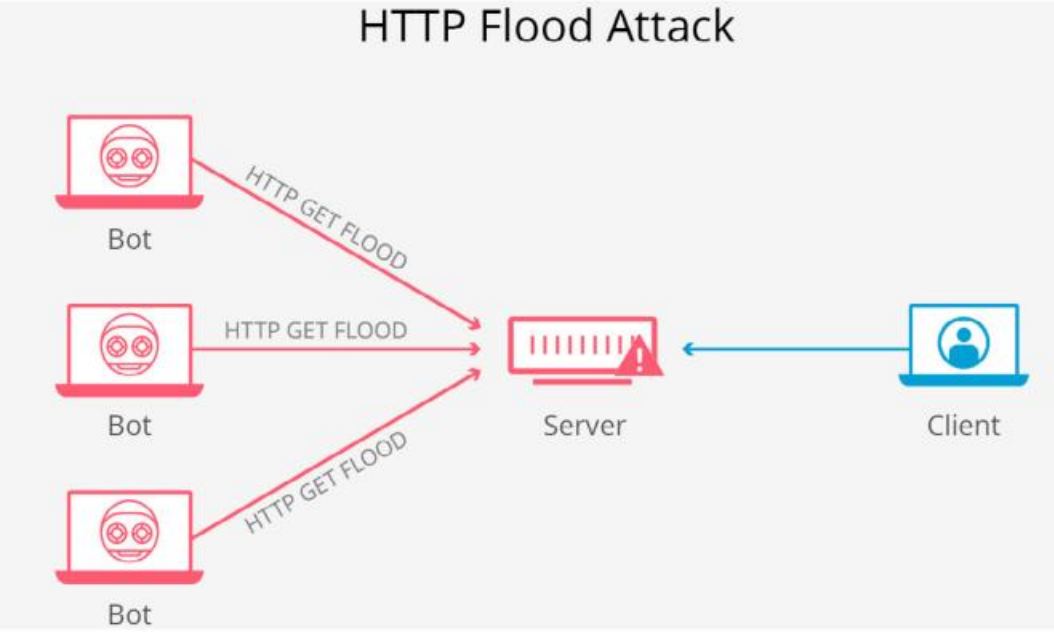



Fig. 10 Flooding attack

**Blackhole**: In this attack (see Figure 11), the system loses some or all of the data that passes through a certain node.

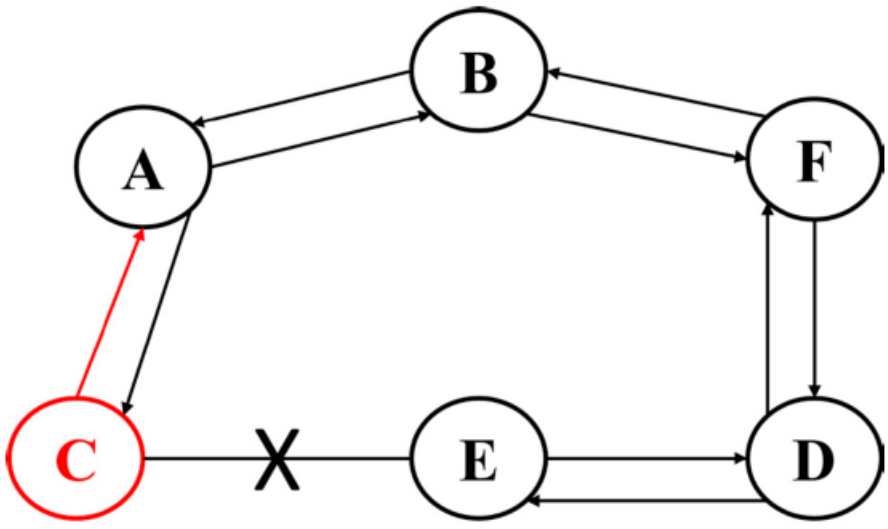



Fig. 11 Blackhole attack

**Grayhole**: This is a more advanced type of Blackhole attack. It usually happens only in wireless sensor networks.

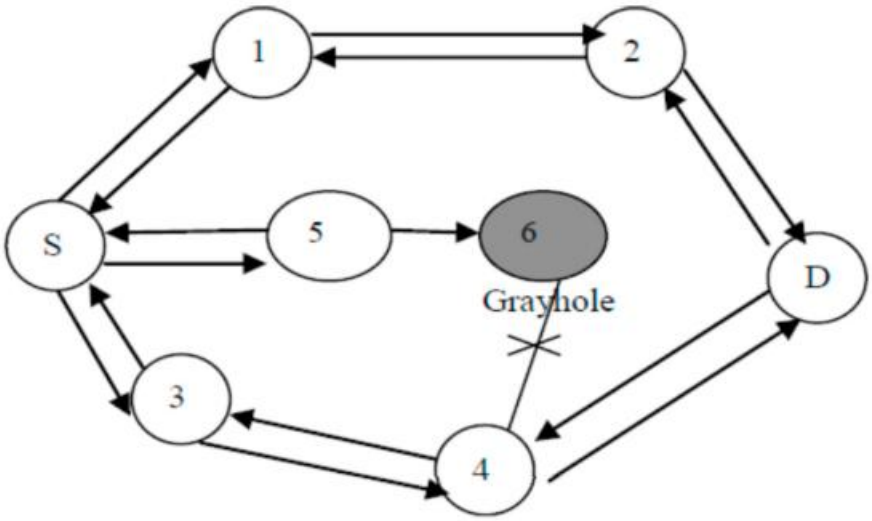



Fig. 12 Grayhole attack

**TDMA**: In this attack, the hacker uses the system's slow clock or timing to check how busy the CPU is. The attack is timed so it avoids getting noticed when the system checks the schedule (Fares et al., 2025).

Table 1 Feature explanation of WSN-DS dataset

| Feature number | Feature symbol | Feature name | Description |
|---|---|---|---|
| 1 | **id** | Node Id | A unique ID number of the sensor node |
| 2 | **Time** | Time | The run-time of the node in the simulation |
| 3 | **Is_CH** | Is CH | Describes if the node is a CH (cluster head) or not |
| 4 | **who CH** | Who CH | Cluster head ID |
| 5 | **Dist_To_CH** | Distance to CH | Distance between node and CH |
| 6 | **ADV_S** | ADV CH sends | Number of the advertise CH's broadcast messages sent to nodes |
| 7 | **ADV_R** | ADV CH receives | Number of advertise messages received by the nodes from CH |
| 8 | **JOIN_S** | Join request send | Number of join request messages sent by the nodes to the CH |
| 9 | **JOIN_R** | Join request receive | Number of join request messages received by CH from nodes |
| 10 | **SCH_S** | ADV SCH sends | Messages of TDMA schedule broadcast sent to the nodes |
| 11 | **SCH_R** | ADV SCH receives | Number of scheduled messages received by the CH |

| | | | |
|---|---|---|---|
| 12 | **Rank** | Rank | Node order in TDMA scheduling |
| 13 | **DATA_S** | Data sent | Number of data packets sent from the node to its CH |
| 14 | **DATA_R** | Data received | Number of data packets received by the node from the CH |
| 15 | **Data_Sent_To_BS** | Data sent to BS (Base station) | Number of data packets that are sent from node to the BS |
| 16 | **dist_CH_To_BS** | Distance CH to BS | Distance between CH and BS |
| 17 | **send_code** | Send code | The sending code of the cluster |
| 18 | **Consumed Energy** | Energy consumption | Energy consumed |
| 19 | **Attack type** | Attack type | Type of attacks or normal traffic |

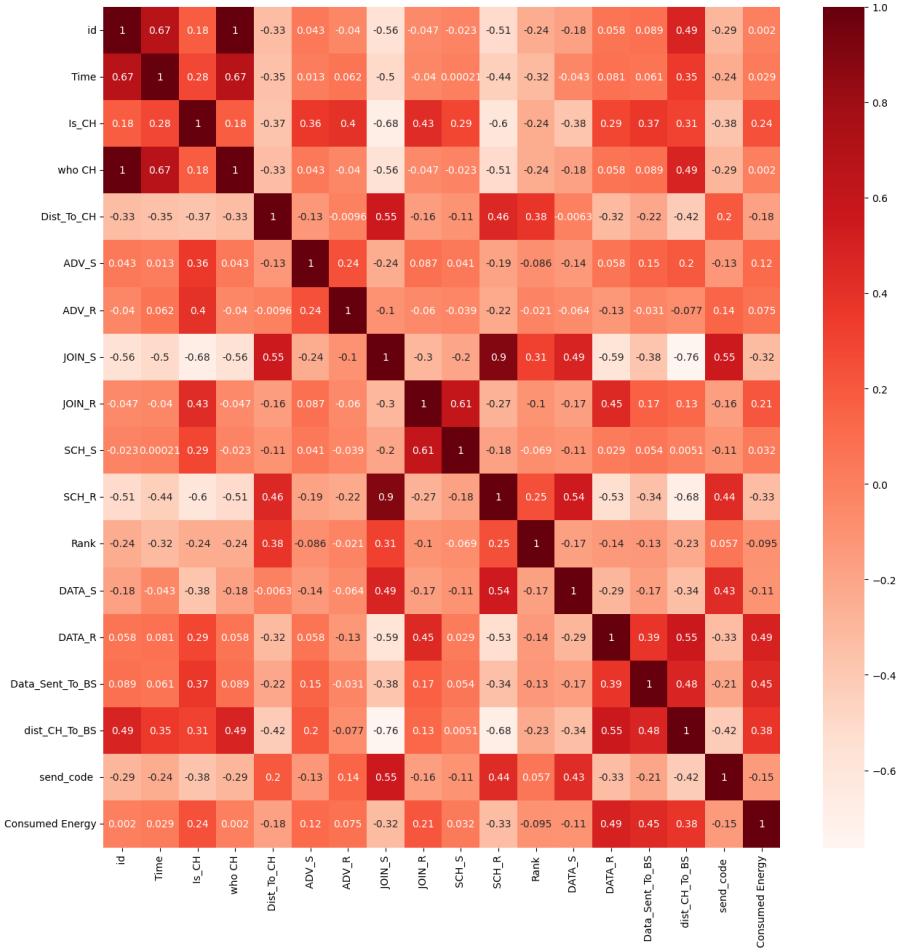
Each record represents network behaviour captured during simulations of different DoS attacks. The features cover routing activity, signal quality, energy levels, and other metrics commonly used to monitor network security in wireless sensor networks.

A key challenge with the dataset is its highly imbalanced class distribution, with the majority of the records belonging to the normal class. This imbalance affects model training and requires appropriate sampling methods to ensure all attack types are properly learned and detected. The labelled structure of the dataset made it suitable for supervised learning tasks in machine learning.

### 5.3 Data Cleaning and Feature Engineering

The Initial data cleaning involved checking for missing or duplicate entries. The dataset was generally clean, so minimal correction was needed. Focus was then placed on preparing the data for modelling by addressing class imbalance and reducing irrelevant or noisy features. Firstly, a correlation matrix was calculated for the dataset features, excluding the 'Attack type' column. This helps to identify which features are strongly related to each other. The correlation heatmap is shown in the Fig below.

Fig. 13 Correlation matrix of the dataset features




ADASYN was used for oversampling which balanced the dataset. It was chosen over SMOTE due to its ability to handle overlapping class boundaries and strong imbalance, especially for rare attack types like TDMA and Flooding. The application ADASYN improved the dataset's quality for learning.

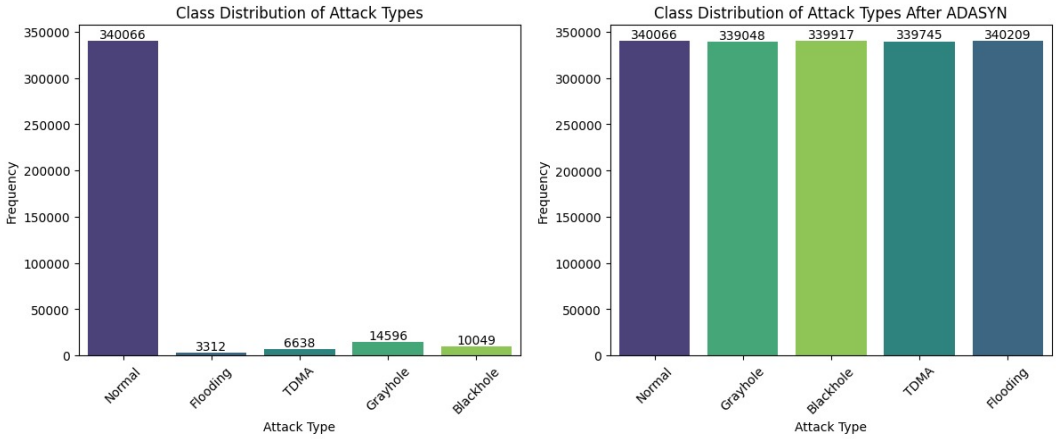



Fig. 14 Class distribution of attack type before and after ADASYN

Mutual information was used for feature selection, which gave a score to each feature based on its importance. These scores are shown in a bar chart in Fig. Based on the scores, the top 13 features were selected to start model training.

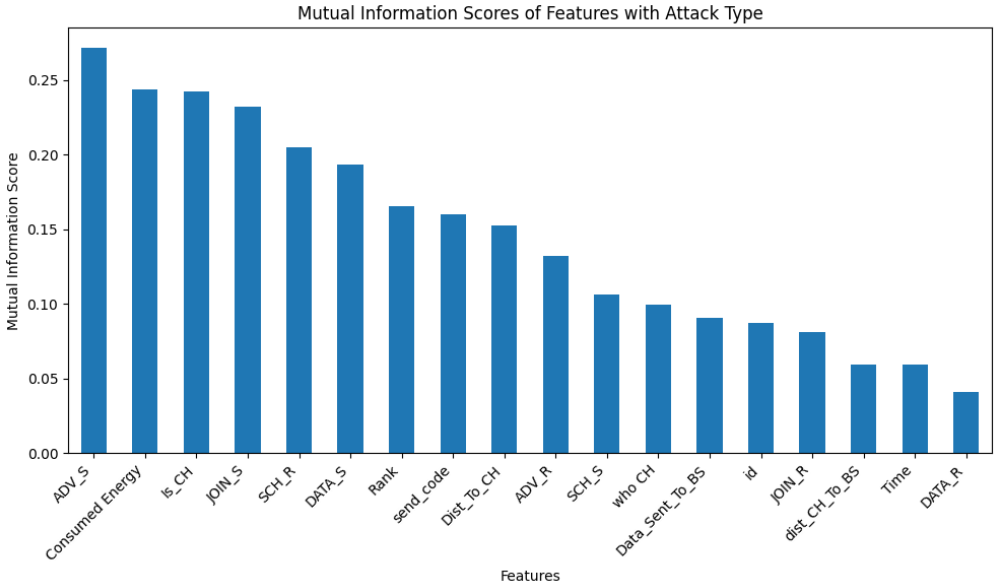



Fig. 15 Bar chart of mutual information score

To improve the trained model performances, feature engineering through PCA (Principal Component Analysis) for dimensionality reduction was applied to reduce the number of features. This reduced the dataset to 9 key features while keeping 95% of the original data variance. The models were then retrained.

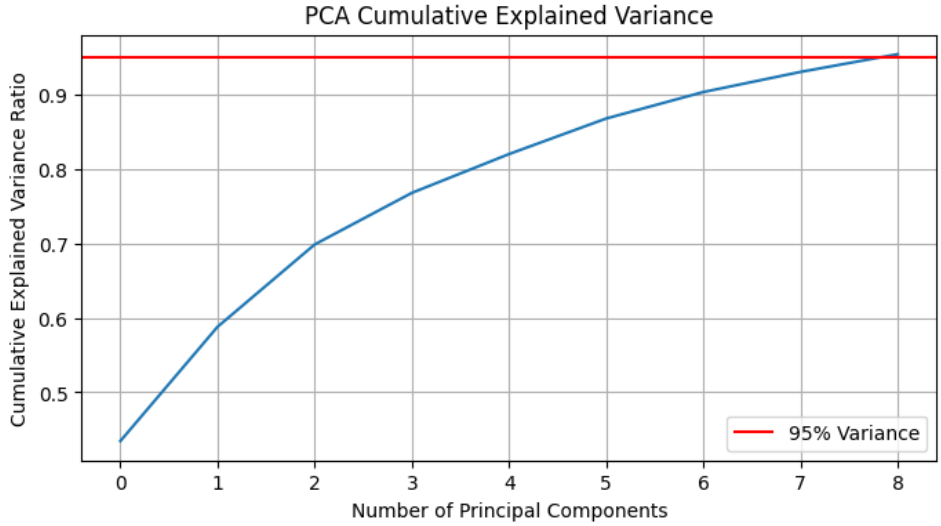



Fig. 16 Title the explained variance ratio plot

Furthermore, to further enhance the model results, LDA (Linear Discriminant Analysis) was used for another round of dimensionality reduction, Fig. shows the scatter plot of data distribution of PCA and LDA. This reduced the features to 4 of the most important ones. The models were then retrained one final time.

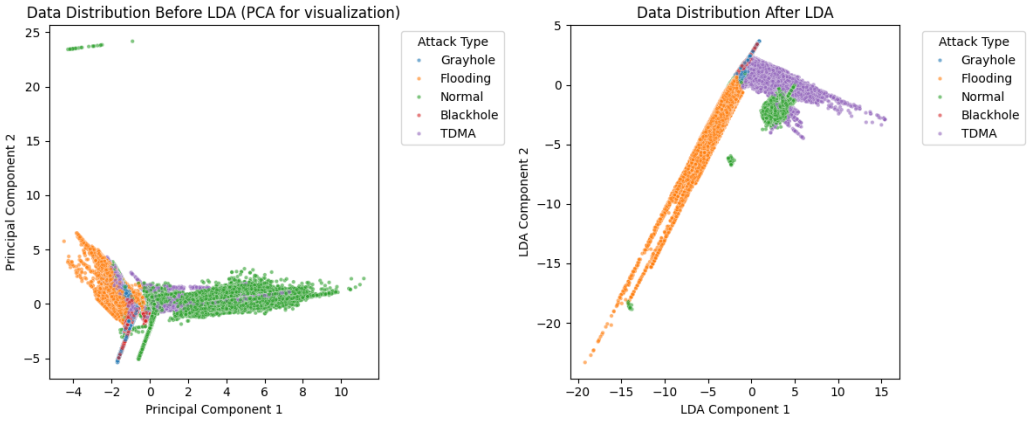



Fig. 17 Data distribution of PCA and LDA

### 5.4 Dataset initialization and Model Training

After applying ADASYN oversampling to balance the dataset, it was split into 70% training and 30% testing sets. Both sets were then scaled (Fig. 2). After scaling, the selected models were trained (XGBoost (XGB), Decision tree, ANN, and CNN) with the processed dataset.

Fig. 18 Fitting the processed dataset for model training

Now we proceed with the scaling and train-test split

```
[ ]  # Define feature set (X) and target (y) using the resampled data
     X = X_resampled
     y = y_resampled

[ ]  #Split the balanced data into training and testing sets using train_test_split.
     x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) #Train-Test Split (70-30 split)

[ ]  #Feature Scaling using StandardScaler
     scaler = StandardScaler()

[ ]  # Fit the scaler on training data and transform both train and test sets
     X_train_scaled = scaler.fit_transform(x_train)
     X_test_scaled = scaler.transform(x_test)
```

## 6.0 Model Evaluation and Results

This section presents the evaluation of the trained models using performance metrics and shows the results, which helped to understand how well each model performed when trained with the dataset in different instances.

### 6.1 Evaluation Methodology

The trained Models performances were evaluated using multiple metrics which include detection accuracy and detection time.

- **Confusion Matrix:** This is a method used to represent the value of the predicted class relative to the actual class, where the diagonal value represents True Positive and Treu Negative respectively, while the other cells represent False Positive and False Negative.
- **Accuracy:** This is the metrics used for measuring the overall precision of the model. The equation below is used to calculate accuracy.

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

*Note: TN – True Negative, TP – True Positive, FP – False Positive, FN – False Negative*

- **Recall:** This is a metrics used to access a trained model capability to capture all the appropriate samples which is calculated with the equation below

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

- **Precision:** Precision is determined accurately by the proportion of labels to all positive classification which is calculated with the formula below

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

- **F1 Score:** It is used to balance precision and recall into one single measure. It is calculated with the formula below

$$F1 - Sccore = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Moreover, Accuracy was used to measure the proportion of correct predictions, serving as the primary metric for model comparison. Furthermore,
Since the dataset was imbalanced, accuracy alone was not sufficient. Therefore, detection time was recorded to assess how fast each model responded, which is critical in real-time intrusion detection systems.

Subsequently, confusion matrix, precision, recall, and F1-score were also utilised to evaluate the trained model's ability to correctly classify minority attack classes. These metrics helped verify that the various applied strategies were effective. However, final model selection

could be based on combined metrics with detection being the focus, ensuring the models can be deployed in practical, time-sensitive environments.

### 6.2 Model Performance Results

After training the models multiple times with different feature engineering, the table below shows the result from each training.

Table 2 Comparison of trained models

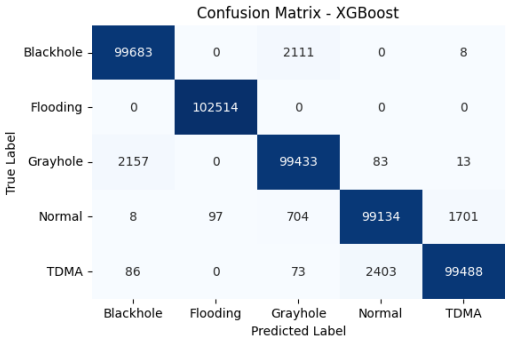
| SN | Model | Accuracy (%) | Prediction Time (s) | Feature Engineering |
|---|---|---|---|---|
| **1** | XGBoost | 0.981471 | 3.415807 | ADASYN |
| **2** | Decision Tree | 0.991766 | 0.094809 | ADASYN |
| **3** | ANN | 0.936227 | 18.529168 | ADASYN |
| **4** | CNN | 0.91097 | 20.532613 | ADASYN |
| **5** | ANN | 0.93647 | 16.245938 | PCA |
| **6** | CNN | 0.921722 | 20.537396 | PCA |
| **7** | XGBoost | 0.947657 | 4.081607 | PCA |
| **8** | Decision Tree | 0.978473 | 0.089592 | PCA |
| **9** | ANN | 0.915801 | 16.775432 | LDA |
| **10** | CNN | 0.911439 | 20.534322 | LDA |
| **11** | XGBoost | 0.928143 | 3.625726 | LDA |
| **12** | Decision Tree | 0.966405 | 0.100984 | LDA |

**XGBoost Model Performance**

The XGBoost model showed a strong result in detecting different attack types and normal traffic, the model achieved an accuracy of 98.1 percent with a prediction time of 3.41 seconds. The Precision, recall, and F1-score were all close to 0.98 across most classes, with Flooding detection reaching 1.00 for all three measures.

The confusion matrix shows minor misclassifications, such as Grayhole being predicted as Blackhole and Normal being confused with TDMA. Despite this, the model maintained a balanced performance in all categories which shows its reliability for intrusion detection.

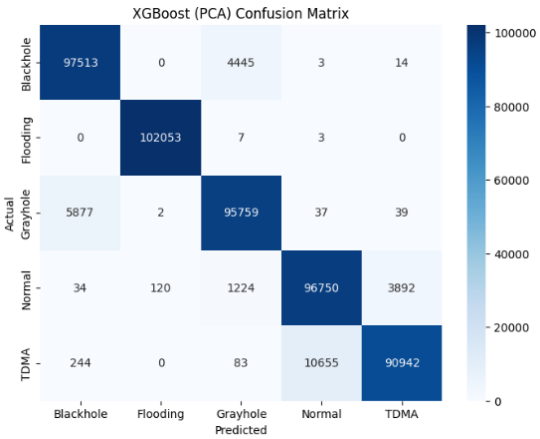
Fig. 19 XGB Confusion matrix and class report of initial training




```
Accuracy: 0.9814713083877449
XGBoost Prediction Time: 3.4158072471618652

Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.98      0.98      0.98    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.97      0.98      0.97    101686
      Normal       0.98      0.98      0.98    101644
        TDMA       0.98      0.97      0.98    102050

    accuracy                           0.98    509696
   macro avg       0.98      0.98      0.98    509696
weighted avg       0.98      0.98      0.98    509696
```

The XGBoost model with PCA achieved an accuracy of 94.8 percent and a prediction time of 4.08 seconds. Flooding detection remained perfect with precision, recall, and F1-score of 1.00. Most classes scored above 0.94 for all metrics, though Normal and TDMA showed reduced precision and recall. The results indicate that PCA preserved high model performance while slightly reducing accuracy compared to first XGBoost result. Misclassifications mainly occurred between Normal and TDMA, showing some overlap after dimensionality reduction.

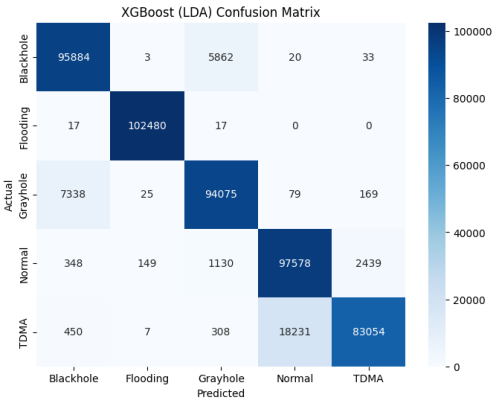
Fig. 20 XGB Confusion matrix and class report of PCA applied training




```
XGBoost (PCA) Accuracy on Test Set: 0.9477

XGBoost (PCA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.94      0.96      0.95    101975
    Flooding       1.00      1.00      1.00    102063
    Grayhole       0.94      0.94      0.94    101714
      Normal       0.90      0.95      0.92    102020
        TDMA       0.96      0.89      0.92    101924

    accuracy                           0.95    509696
   macro avg       0.95      0.95      0.95    509696
weighted avg       0.95      0.95      0.95    509696
```

The XGBoost model with LDA achieved an accuracy of 92.8 percent with a prediction time of 3.63 seconds. Flooding detection again remained perfect, but the Normal and TDMA classes showed weaker performance. Normal traffic achieved high recall (0.96) but lower precision (0.84), while TDMA had strong precision (0.97) but lower recall (0.81).

Fig. 21 XGB Confusion matrix and class report of LDA applied training




```
XGBoost (LDA) Prediction Time: 3.6257 seconds

XGBoost (LDA) Accuracy on Test Set: 0.9281

XGBoost (LDA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.92      0.94      0.93    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.93      0.93      0.93    101686
      Normal       0.84      0.96      0.90    101644
        TDMA       0.97      0.81      0.88    102050

    accuracy                           0.93    509696
   macro avg       0.93      0.93      0.93    509696
weighted avg       0.93      0.93      0.93    509696
```

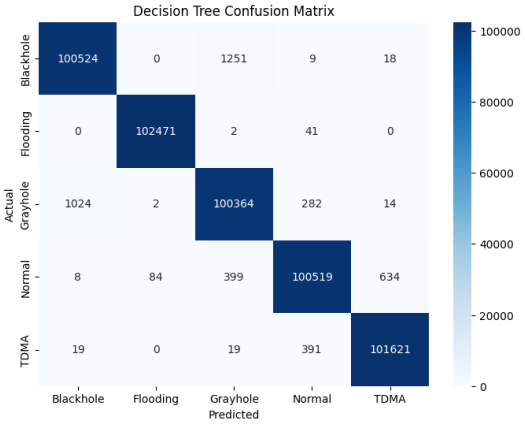
When compared to PCA trained model, the LDA version showed higher class imbalance effects, even though the performance was still strong, the drop in precision and recall across Normal and TDMA indicates that applying LDA reduced the overall balance of the model.

**Decision Tree Model Performance**

The initial training of the Decision Tree model achieved an accuracy of 99.2 percent with a prediction time of 0.09 seconds, making it efficient for real-time detection. Precision, recall, and F1-scores were all close to 0.99 for every class, with Flooding detection again reaching 1.00.

The confusion matrix shows small errors, such as Grayhole being predicted as Blackhole and Normal misclassified as TDMA. Despite these misclassifications, the overall results confirm the model's strong and balanced performance across all attack types and normal traffic.

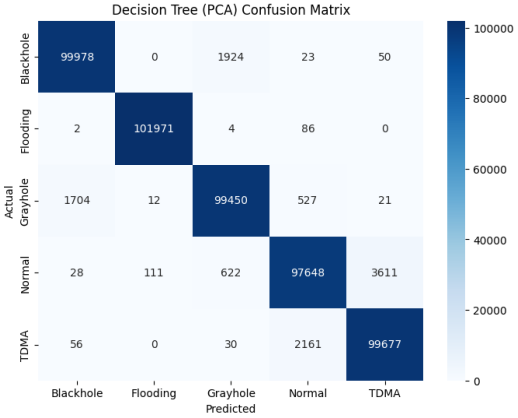
Fig. 22 DT Confusion matrix and class report of initial training




```
Decision Tree Accuracy on Test Set: 0.9918

Decision Tree Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.99      0.99      0.99    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.98      0.99      0.99    101686
      Normal       0.99      0.99      0.99    101644
        TDMA       0.99      1.00      0.99    102050

    accuracy                           0.99    509696
   macro avg       0.99      0.99      0.99    509696
weighted avg       0.99      0.99      0.99    509696
```

The Decision Tree model trained with PCA achieved an accuracy of 97.9 percent and a fast prediction time of 0.09 seconds. Performance remained strong across all classes, with precision, recall, and F1-scores around 0.97 to 0.98. Flooding detection again achieved perfect results. When compared to the initial trained Decision Tree model, the PCA trained version showed a slight drop in accuracy but maintained balanced performance across all classes. This indicates that PCA reduced data dimensions without seriously affecting the detection quality.

Fig. 23 DT Confusion matrix and class report of PCA applied training




```
Decision Tree (PCA) Accuracy on Test Set: 0.9785

Decision Tree (PCA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.98      0.98      0.98    101975
    Flooding       1.00      1.00      1.00    102063
    Grayhole       0.97      0.98      0.98    101714
      Normal       0.97      0.96      0.96    102020
        TDMA       0.96      0.98      0.97    101924

    accuracy                           0.98    509696
   macro avg       0.98      0.98      0.98    509696
weighted avg       0.98      0.98      0.98    509696
```

The Decision Tree model with LDA trained data achieved an accuracy of 96.6 percent and a prediction time of 0.10 seconds. Flooding detection remained perfect, but Normal and TDMA showed slightly lower recall and precision compared to the PCA version.

Fig. 24 DT Confusion matrix and class report of LDA applied training

```
Decision Tree (LDA) Accuracy on Test Set: 0.9664

Decision Tree (LDA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.97      0.97      0.97    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.96      0.96      0.96    101686
      Normal       0.96      0.94      0.95    101644
        TDMA       0.95      0.96      0.96    102050

    accuracy                           0.97    509696
   macro avg       0.97      0.97      0.97    509696
weighted avg       0.97      0.97      0.97    509696
```

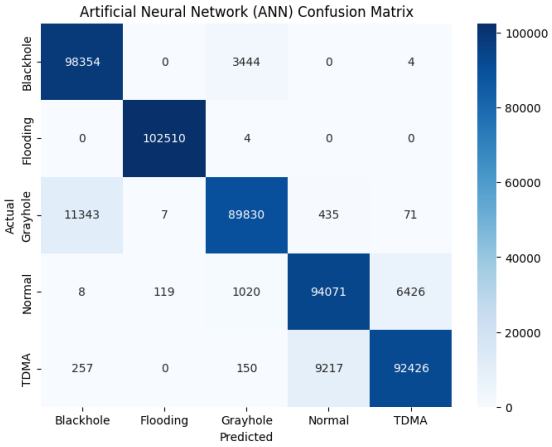
Though the model is efficient and reliable, the LDA-based model underperformed compared to the PCA-based one. This suggests PCA was more effective than LDA in maintaining Decision Tree performance during dimensionality reduction.

**Artificial Neural Network (ANN) Model Performance**

The initial ANN model achieved an accuracy of 93.6 percent with a prediction time of 18.53 seconds. While Flooding detection remained perfect with precision, recall, and F1-score of 1.00, other classes showed weaker performance.

The confusion matrix highlights notable misclassifications. Many Grayhole samples were detected as Blackhole, and many Normal instances were confused with TDMA.

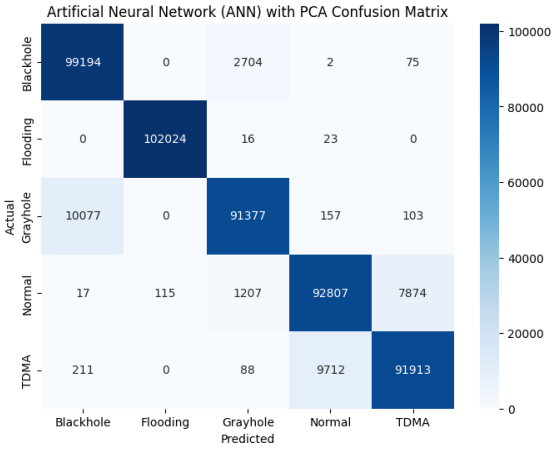
Fig. 25 ANN Confusion matrix and class report of initial training




```
ANN Accuracy on Test Set: 0.9362

ANN Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.89      0.97      0.93    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.95      0.88      0.92    101686
      Normal       0.91      0.93      0.92    101644
        TDMA       0.93      0.91      0.92    102050

    accuracy                           0.94    509696
   macro avg       0.94      0.94      0.94    509696
weighted avg       0.94      0.94      0.94    509696
```

Furthermore, the Artificial Neural Network (ANN) was retrained using two dimensionality reduction methods, Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) for improved result.

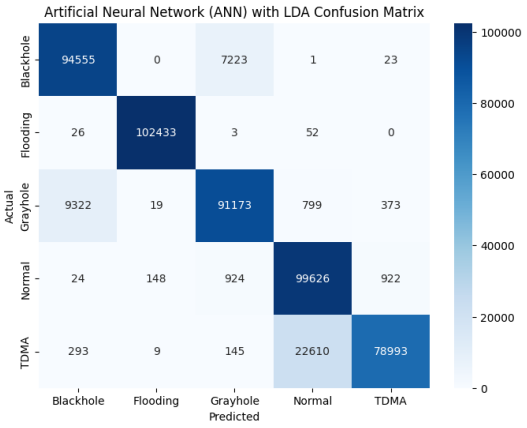
Fig. 26 ANN Confusion matrix and class report of PCA applied training




```
ANN (PCA) Accuracy on Test Set: 0.9365

ANN (PCA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.91      0.97      0.94    101975
    Flooding       1.00      1.00      1.00    102063
    Grayhole       0.96      0.90      0.93    101714
      Normal       0.90      0.91      0.91    102020
        TDMA       0.92      0.90      0.91    101924

    accuracy                           0.94    509696
   macro avg       0.94      0.94      0.94    509696
weighted avg       0.94      0.94      0.94    509696
```

The ANN trained with PCA achieved an accuracy of 93.65 percent, while the ANN with LDA achieved 91.58 percent. The PCA trained model produced a higher precision and recall in most classes, especially for Blackhole and TDMA. The LDA trained model, although slightly weaker in overall accuracy, provided strong recall for Normal traffic with 0.98.

Fig. 27 ANN Confusion matrix and class report of LDA applied training

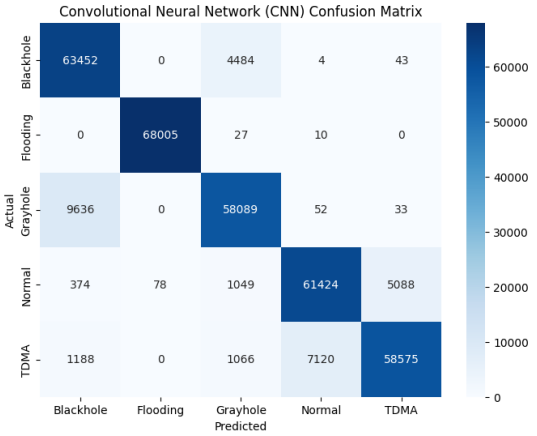



```
ANN (LDA) Prediction Time: 16.7754 seconds

ANN (LDA) Accuracy on Test Set: 0.9158

ANN (LDA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.91      0.93      0.92    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.92      0.90      0.91    101686
      Normal       0.81      0.98      0.89    101644
        TDMA       0.98      0.77      0.87    102050

    accuracy                           0.92    509696
   macro avg       0.92      0.92      0.92    509696
weighted avg       0.92      0.92      0.92    509696
```

The prediction time was close for both methods, with PCA at 16.25 seconds and LDA at 16.77 seconds. The results suggest that PCA preserved more discriminative features for classification, while LDA performed well for certain traffic types but struggled with TDMA misclassifications. Consequently, these results indicate that the PCA trained ANN is the better choice in this dataset due to the balanced performance and higher overall accuracy.
Convolutional Neural Network with LDA and PCA

The Convolutional Neural Network (CNN) was trained with sampled input, LDA, and PCA feature reduction. All three models classified five categories of traffic: Blackhole, Flooding, Grayhole, Normal, and TDMA with significantly low false positive misclassification. The initial trained CNN model reached 91.10 percent accuracy with a balanced performance across most classes.

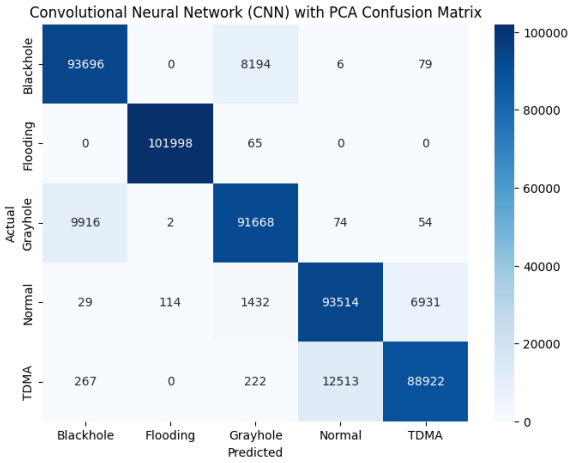
Fig. 28 CNN Confusion matrix and class report of initial training




```
CNN Prediction Time: 20.5326 seconds

CNN Accuracy on Test Set: 0.9110

CNN Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.85      0.93      0.89     67983
    Flooding       1.00      1.00      1.00     68042
    Grayhole       0.90      0.86      0.88     67810
      Normal       0.90      0.90      0.90     68013
        TDMA       0.92      0.86      0.89     67949

    accuracy                           0.91    339797
   macro avg       0.91      0.91      0.91    339797
weighted avg       0.91      0.91      0.91    339797
```

The PCA trained CNN model improved accuracy slightly to 92.17 percent with a higher precision for Blackhole and TDMA while maintaining perfect detection for Flooding.

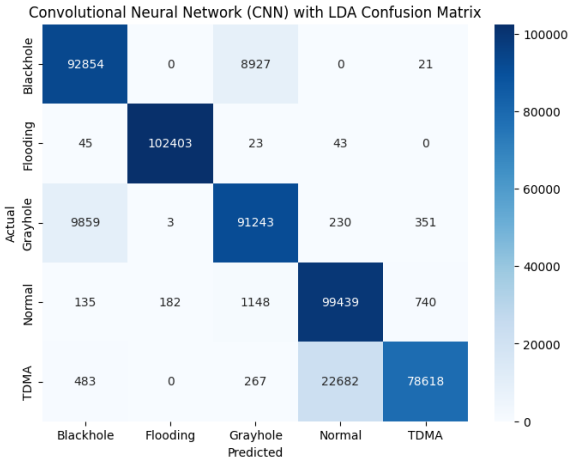
Fig.29 CNN Confusion matrix and class report of PCA applied training




```
CNN (PCA) Accuracy on Test Set: 0.9217

CNN (PCA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.90      0.92      0.91    101975
    Flooding       1.00      1.00      1.00    102063
    Grayhole       0.90      0.90      0.90    101714
      Normal       0.88      0.92      0.90    102020
        TDMA       0.93      0.87      0.90    101924

    accuracy                           0.92    509696
   macro avg       0.92      0.92      0.92    509696
weighted avg       0.92      0.92      0.92    509696
```

The CNN trained with LDA reduced feature achieved 91.14 percent accuracy, performing strongly for Normal traffic with a recall of 0.98 but showing weaker results in TDMA misclassifications.

Fig. 30 CNN Confusion matrix and class report of LDA applied training




```
CNN (LDA) Accuracy on Test Set: 0.9114

CNN (LDA) Classification Report:
              precision    recall  f1-score   support

   Blackhole       0.90      0.91      0.91    101802
    Flooding       1.00      1.00      1.00    102514
    Grayhole       0.90      0.90      0.90    101686
      Normal       0.81      0.98      0.89    101644
        TDMA       0.99      0.77      0.86    102050

    accuracy                           0.91    509696
   macro avg       0.92      0.91      0.91    509696
weighted avg       0.92      0.91      0.91    509696
```

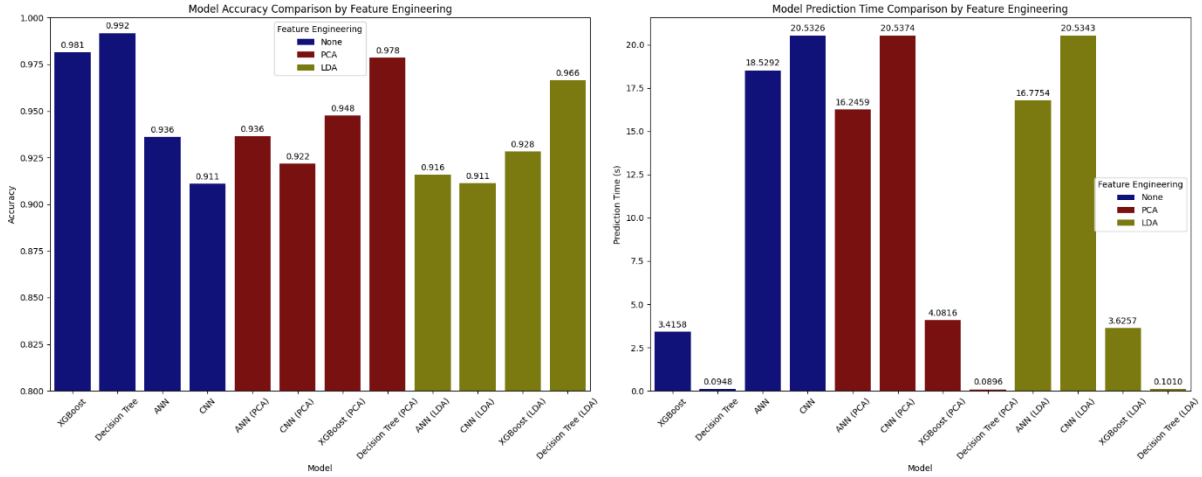
The attack prediction time remained consistent across all models, which averages around 20.5 seconds. The PCA trained model provided the best trade-off between accuracy and class balance, while LDA trained model showed selective strengths but has an overall lower

performance compared to the PCA trained model. These results showed that the PCA trained CNN model achieved the most reliable classification for this dataset.

### 6.3 Comparison of Models with base research paper

A comparison was made between the trained models in this project and the study of Fares et al. (2025), which applied SVM, KNN, and Decision Tree. Although the models in their study achieved high accuracy, the detection times were much slower compared to the results of this project. Their reported testing times were 29 seconds for SVM, 27 seconds for Decision Tree, and 33 seconds for KNN. In contrast, the models from this project achieved faster detection times, making them more efficient for real-world use. The lowest detection time among the trained models in this project, after applying different feature selection and engineering methods, averaged 20.5 seconds. This demonstrates that the models developed in this project are not only accurate but also more practical and suitable for deployment in production environments.

Fig. 31 Bar chart comparison of the accuracy and detection time of all trained models




### 6.4 Discussion of Results vs. Objectives

The project aimed to design a machine learning intrusion detection system that works across TSNs, smart homes, and WSNs and the results from the trained models achieved this aim by showing high accuracy across different models. XGBoost reached 98.1 percent accuracy with strong class balance, especially for Flooding attacks which scored 1.00 in all metrics.

Although application of PCA and LDA reduced the performance slightly but still showed reliable results. Decision Tree gave the highest accuracy of 99.2 percent with fast prediction time of 0.09 seconds, proving efficient for real-time detection. ANN reached 93.6 percent accuracy but required longer prediction time of over 18 seconds, which reduces its real-time suitability. CNN achieved 91.1 percent accuracy, and when trained with PCA, it improved to 92.17 percent with strong precision for difficult classes like TDMA.

The objectives were also addressed, class data imbalance was resolved using ADASYN, which improved minority attack detection. False positives remained low across all the models, with most errors occurring between Normal and TDMA traffic. Model comparison showed Decision Tree best for speed, and XGBoost best for accuracy and balance.

Therefore, the objectives of implementing and evaluating different ML models, addressing imbalance, and balancing accuracy with speed were achieved successfully.

### 6.5 Lessons Learned

The results showed that machine learning can detect complex intrusion attacks effectively, but each algorithm has some trade-offs. Decision Tree was the fastest with 0.09 second prediction time, making it ideal for real-time detection, but it was more sensitive to misclassifications after dimensionality reduction. XGBoost offered the best overall accuracy and class balance, but at a slower speed compared to Decision Tree. ANN and CNN models showed good accuracy but required long prediction times, making them less suitable for low-latency systems.

It was also learned that dimensionality reduction methods affect performance differently, PCA generally preserved classification accuracy better than LDA across models. LDA improved recall in some cases but reduced precision and overall balance. Another key lesson is that misclassification often happened between Normal and TDMA traffic, showing that feature overlap is a challenge. The project highlighted the importance of selecting algorithms that balance accuracy, efficiency, and class fairness.

## 7.0 Conclusions and Future Work

This project designed and tested a machine learning intrusion detection system using datasets from WSN-DS and multiple algorithms were applied and compared. Decision Tree achieved the highest accuracy at 99.2 percent with the fastest speed, proving effective for environments needing low latency. XGBoost achieved 98.1 percent accuracy with balanced detection across all classes, showing strong reliability. ANN and CNN models performed well, but longer prediction times above 16 seconds limited their real-time use. Moreover, application of PCA shown to be more effective than LDA in preserving model balance and accuracy during dimensionality reduction among the trained models.

The research questions were addressed as Machine learning based IDS proved more effective in detecting low-rate and complex attacks. The dataset imbalance had a clear impact, but oversampling and feature reduction addressed the impact and improved results. Trade-offs between accuracy and speed were observed, with Decision Tree being fastest and XGBoost being most balanced. False positives were kept low with proper feature selection and threshold tuning.

Future work should involve testing with new datasets that include modern IoT and TSN attack traffic. Models should be optimised further for speed in resource-limited devices. Deep learning models such as hybrid CNN-LSTM could be explored, along with federated learning for privacy-preserving IDS in smart environments.

### 7.1 Summary of Achievements

The project developed a machine learning intrusion detection system and tested multiple algorithms for accuracy, speed, and false positive rates. Decision Tree achieved the highest accuracy of 99.2 percent with very fast predictions, while XGBoost provided the best class balance at 98.1 percent accuracy. ANN and CNN models also performed well but had higher prediction times, making them less suitable for real-time deployment. PCA was found to preserve accuracy better than LDA in dimensionality reduction. The IDS successfully met its aim of detecting modern attacks with high accuracy and controlled false positives across TSNs, smart homes, and WSNs.

### 7.2 Research Contributions

This project contributed by comparing multiple machine learning models for IDS across TSNs, smart homes, and WSNs. It provided clear evidence that Decision Tree is the best for speed while XGBoost is the best for balanced detection accuracy, It showed how PCA improves model balance better than LDA during dimensionality reduction. The research also highlighted that Normal and TDMA traffic are difficult to separate, which is a key area for further study. The IDS framework developed here can guide future research on balancing speed, accuracy, and false positives for practical use in real-time IoT environments.

### 7.3 Recommendations for Future Improvements

Future improvements should include testing the IDS with newer datasets that capture emerging IoT and TSN attack traffic. Data imbalance should be handled with advanced oversampling methods such as SMOTE-Tomek and hybrid feature selection to improve

classification of minority attacks. Lightweight models need to be developed for resource-limited IoT devices, as ANN and CNN were slightly too slow for real-time use. Deep learning models like CNN-LSTM should be tested to see if they improve temporal attack detection.

To further reduce false positives, ensemble approaches that combine supervised and anomaly-based detection could be applied (Lai et al., 2023). Deployment in real-world testbeds such as smart homes or WSN labs should be done to measure system performance under realistic network conditions. Federated learning should also be explored to train models across multiple devices without centralising sensitive data. These steps would improve accuracy, efficiency, and reliability of the IDS for a practical cyber defence (N et al., 2023).

## 7.0 References

References
AFROZ, M., NYAKWENDE, E. and GOSWAMI, B., 2025. Intrusion Detection in Smart Home Environments: A Machine Learning Approach. *Transportation Research Procedia,* 84, pp. 243–250.
ALI, A.H. et al., 2024. Unveiling machine learning strategies and considerations in intrusion detection systems: a comprehensive survey. *Frontiers in Computer Science,* 6, pp. 1387354.
ALMOMANI, I., AL-KASASBEH, B. and AL-AKHRAS, M., 2016. WSN-DS: a dataset for intrusion detection systems in wireless sensor networks. *Journal of Sensors,* 2016(1), pp. 4731953.
FARES, H. et al., 2025. Intrusion Detection in Wireless Sensor Networks using Machine Learning. *Procedia Computer Science,* 252, pp. 912–921.
FINN, M. and SHILTON, K., 2023. Ethics governance development: The case of the Menlo Report. *Social Studies of Science,* 53(3), pp. 315–340.
HAGERTY, A. and RUBINOV, I., 2019. Global AI ethics: a review of the social impacts and ethical implications of artificial intelligence. *arXiv preprint arXiv:1907.07892,*
JIHADO, A.A. and GIRSANG, A.S., 2024. Hybrid deep learning network intrusion detection system based on convolutional neural network and bidirectional long short-term memory. *J.Adv.Inform.Technol,* 15(2), pp. 219–232.
NEUPANE, S. et al., 2022. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access,* 10, pp. 112392–112415.
NTOUTSI, E. et al., 2020. Bias in Data-driven AI Systems -
An Introductory Survey.
PETROSYAN, A., 2025. *Estimated annual cost of cybercrime in the United Kingdom (UK) from 2017 to 2028.* [online]. Available from: https://www.statista.com/forecasts/1425776/uk-cybercrime-cost-annual
TOPSAKAL, M., CEVHER, S. and ERGENÇ, D., 2025. A machine learning-based intrusion detection framework with labeled dataset generation for IEEE 802.1 time-sensitive networking. *Journal of Systems Architecture,* , pp. 103408.
VOURGANAS, I.J. and MICHALA, A.L., 2024. Applications of machine learning in cyber security: a review. *Journal of Cybersecurity and Privacy,* 4(4), pp. 972–992.
The Models
AHMAD, Z. et al., 2021. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies,* 32(1), pp. e4150.
AL-AMBUSAIDI, M. et al., 2024. ML-IDS: an efficient ML-enabled intrusion detection system for securing IoT networks and applications. *Soft Computing,* 28(2), pp. 1765–1784.
ALHASAN, R.A. and HAMZA, E.K., 2023. A Novel CNN Model with Dimensionality Reduction for WSN Intrusion Detection. *Revue d'Intelligence Artificielle,* 37(5),
FARES, H. et al., 2025. Intrusion Detection in Wireless Sensor Networks using Machine Learning. *Procedia Computer Science,* 252, pp. 912–921.
IBM, a. What is a decision tree?
IBM, b. *What is a neural network?* [online]. Available from: https://www.ibm.com/think/topics/neural-networks
JINGJING, Z. et al., 2022. Intrusion Detection Model for Wireless Sensor Networks Based on MC-GRU. *Wireless Communications and Mobile Computing,* 2022(1), pp. 2448010.
LAI, T.T. et al., 2023. DoS attack detection using online learning techniques in wireless sensor networks. *Alexandria Engineering Journal,* 85, pp. 307–319.

N, D. et al., 2023. Intrusion Detection in Novel WSN-Leach Dos Attack Dataset using Machine Learning based Boosting Algorithms. *Procedia Computer Science,* 230, pp. 90–99.
PANDEY, V.K. et al., 2025. Enhancing intrusion detection in wireless sensor networks using a Tabu search based optimized random forest. *Scientific Reports,* 15(1), pp. 18634.