INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING

BY

TAN MAY MAY

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING

Academic Session: JAN 2024

I TAN MAY MAY

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible] [illegible]

(Author's signature) (Supervisor's signature)

Address:
26, JALAN SEKSYEN U13/42J,
SEKSYEN U13, SETIA ALAM,
40170 SHAH ALAM, SELANGOR.

CIK ZANARIAH BINTI ZAINUDIN
Supervisor's name

Date: 23/4/24 **Date**: 23/4/24

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: __23/4/24_________

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ______***Tan May May***________________ (ID No: __***20ACB01863***__) has completed this final year project entitled "_______*Intusion Detection System (IDS) using Machine Learning*____ _" under the supervision of ___Cik Zanariah binti Zainudin_____ (Supervisor) from the Department of __Digital Economy Technology__, Faculty of _Information and Communication Technology_.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

[illegible]

____
(*Tan May May*)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	Tan May May
Date	:	23/4/24

ACKNOWLEDGEMENTS

I would like to express thanks and appreciation to my supervisor, Cik Zanariah binti Zainudin and my moderator, Ts. Dr. Gan Ming Lee who have given me a golden opportunity to involve in the machine learning and deep learning field study. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

Finally, I must say thanks to my parents and my friends for their love, support, and continuous encouragement throughout the course.

ABSTRACT

Intrusion Detection Systems (IDS) play a critical role in safeguarding organizational networks by identifying potential threats and anomalies. However, traditional IDS approaches often suffer from high false alarm rates, leading to unnecessary alerts for administrators. This research focuses on enhancing machine learning and deep learning-based IDS models to improve accuracy, precision, recall, and F1 score, ultimately aiming for a more balanced and effective performance. The study leverages the CIC-IDS2017 dataset for model training, employing Random Forest (RF), Deep Neural Network (DNN), and Deep Autoencoder (DAE) architectures. A rigorous pre-processing phase, including data cleaning and feature selection using Pearson’s Correlation, enhances the dataset's quality and relevance. Subsequently, the refined dataset undergoes model training and testing. Hyperparameter tuning, facilitated by grid search, fine-tunes key features to optimize model performance. Evaluation metrics such as accuracy, precision, recall, and F1 score are employed to assess the models' efficacy in binary and multi-class classification tasks. Results demonstrate significant improvements and balanced performance compared to previous research models, achieving an average performance of 99.5% across all models.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IDS	Intrusion Detection System
RF	Random Forest
DNN	Deep Neural Network
DAE	Deep Autoencoder
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
DoS	Denial of Service
DDoS	Distributed Denial of Service

Chapter 1
Introduction

An intrusion detection system (IDS) is a kind of technology that assists in identifying potential security flaws in a target application or computer and works to protect that system. Many people find the detection useful in identifying potential viruses and assaults. But studies have shown that most intrusion detection systems (IDS) could produce more false positives when there isn't any suspicious network activity [1]. The administrators receive an unwarranted alert as a result of this. Thus, by using hyperparameter tuning to the existing machine learning models for IDS, the research aims to enhance the precision and accuracy of the IDS while simultaneously reducing the number of needless warnings and raising the detection rate.

1.1 Problem Statement and Motivation

The volume of data has increased throughout time, exhibiting a variety of formats and structure combinations. As a result, a dataset that is used to train models is usually unstructured, inconsistent, and filled with redundant data. This is because noise, outliers, and missing values in raw data can all harm the model's performance by causing the model to overfitting. Pre-processing methods like noise reduction, outlier identification and removal, and replacement for missing values will help ensure the training data is clean and dependable, resulting in more precise predictions and improved performance overall [2].

Performance measures including accuracy, precision, recall, F1 score, and confusion matrix are critical tools for conducting a thorough evaluation and comparing each model's performance across many prediction quality aspects. These measures play a crucial role in assessing models from various angles, including their ability to correctly detect occurrences as opposed to their tendency to produce false positives [3]. By using these measures effectively, the research may assess models' effectiveness and determine which ones perform better, helping to investigate the most appropriate model for different use cases and requirements.

As a result of the high detection rates of IDS using machine learning, several machine learning models were investigated for possible application in IDS. Nevertheless, studies have shown that these models are imperfect and can be improved in terms of detection quality, including precision and accuracy [4]. As a result, improving the quality of IDS models is necessary, and one feasible method of doing it is through hyperparameter tuning. Hyperparameter tuning attempts to find the ideal model configuration, improving performance and reducing the amount of time organizations need to discover which parameters work best for the selected models.

With the increasing of machine learning models introduced to the research world, this research aims to increase the quality of machine learning models in the perspective of accuracy, precision, recall, and F1 score. This work aims to facilitate the utilization of existing machine learning models introduced by the researchers with improved quality. This is accomplished using the hyperparameter research conducted. This approach intends to accelerate the hyperparameter research process, allowing companies that used the models to spend less time exploring hyperparameters and more time analysing results and deriving value from the models.

1.2 Objectives

1. To implement data pre-processing and feature selection for better performance and accuracy of machine learning models in later processes.
2. To analyse each machine learning model's performance using parameters such as accuracy, precision, recall, and f1 score.
3. To optimize the machine learning models by hyperparameter tuning the models to increase their accuracy, precision, recall, and f1 score.

This research paper's main objectives have been divided into three parts. First and foremost, the study intends to apply data pre-processing and feature selection methods to the dataset. In order to guarantee accurate and objective training outcomes, it is essential to take this step before training different machine learning models to ensure the structural quality of the data.

Second, a thorough performance evaluation of the trained machine learning models was conducted using industry-standard metrics like accuracy, precision, recall, and F1 score. This evaluation technique is necessary to fairly compare the models by objectively evaluating each model's performance from many angles.

Finally, to further optimize the trained machine learning models, hyperparameter tuning is used in the study. The objective is to determine which hyperparameters are able to improve the models' recall, accuracy, precision, and F1 score. The research intends to improve the performance of existing machine learning models developed by researchers by modifying hyperparameters, resulting in increased overall model efficacy and predictive capacities.

1.3 Project Scope and Direction

The research covers several scopes to achieve the objectives and address the problem statements. These scopes are pre-processing the dataset and conducting feature selection, training and testing the dataset using the chosen models, hyperparameter tuning the models, and comparing the performance of models before and after hyperparameter tuning with results from previous studies.

The study will first choose an existing dataset to demonstrate the network traffic consisting of both normal and attack traffic. As collecting network traffic will be time-consuming and requires ensuring it consists of the most important features with various types of attacks, it will not be considered in this research to collect the massive dataset with few Gigabits. Instead, CIC-IDS2017 will be chosen as the dataset to be trained by different models which consist of 14 different attacks including frequent attacks such as Denial of Service (DoS), Distributed Denial of Service (DDoS), and Brute Force [5].

Pre-processing will be applied to the chosen dataset in order to transform the data in a more organized manner. This pre-processing phase includes numerous critical procedures, such as combining data files, dealing with null values and labels, scaling the data, converting categorical data to numerical representation, and maintaining data balance. Feature selection technique has been used to identify and preserve the most relevant characteristics to ensure that the models trained without being impacted by unimportant or irrelevant features.

Three types of deep learning or machine learning models will be chosen as the main models to be trained by examining previous research articles authored by other researchers. Subsequently, the trained models' performances will be assessed using the parameter metrics regarding recall, accuracy, precision, and F1 score. After that, the models will be assessed using the same parameter metrics and hyperparameter adjusted. The outcomes will be recorded and contrasted with the performance outcomes of the previous researchers.

1.4 Contributions

In order to maximize the performance of trained machine learning models, the study describes the critical phases of pre-processing and feature selection using CIC-IDS2017. The study guarantees that the machine learning model training and testing that follows will produce objective results that are in line with the research objectives and problem statements by putting these pre-processing techniques into practice.

In addition, the study presents three models that have optimized hyperparameters, improving upon the models developed by earlier researchers in terms of performance metrics including accuracy, precision, recall, and F1 score. This solution drastically cuts down the amount of time needed for hyperparameter tuning while simultaneously enhancing model performance. Organizations gain from this simplified method since it eases the research to obtain the hyperparameters that improve IDS performance and frees them up to concentrate more on output analysis and business operations.

1.5 Report Organization

This report is organized into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology, Chapter 4 Data Pre-processing, Chapter 5 Modelling and Hyperparameter Tuning, Chapter 6 Experimental Result and Discussion, and Chapter 7 Conclusion. The first chapter is the introduction of this project which includes the problem statement, project background and motivation, project scope, project objectives, project contribution, highlights of project achievements, and report organization. The second chapter is the literature review carried out on several existing machine learning models used in this research with the study of data pre-processing, feature selection techniques, and hyperparameter tuning methods. The third chapter discusses the overall flow and requirements of this research. The fourth chapter is regarding the details on how to implement the pre-processing and feature selection on the dataset chosen. The fifth chapter regarding the details of implementing the modeling steps and hyperparameter tuning on the models chosen. The sixth chapter discussed about the results returned from the fourth and fifth chapters. The seventh chapter concluded the research and results with a summary.

Chapter 2
Literature Review

2.1 Previous Researchers' Work on Different Models

The potential machine learning and deep learning models that could be utilized in an intrusion detection system have been investigated by previous researchers. Three machine learning or deep learning models that have been employed by earlier academics to evaluate performances will be the focus of this research. The paper selected three machine learning and deep learning models: Random Forest, Autoencoders on Deep Learning Model, and Deep Neural Network with numerous hidden layers.

[6] Rifkie Primartha and Bayu Adhi Tama conducted the first review of Random Forest, which focused on its performance in intrusion detection. The NSL-KDD, UNSW-NB15, and GPRS datasets were used in the experiment to train the machine learning model. To confirm the results, the researchers also used a 10-fold cross-validation technique, primarily concentrating on the accuracy and false alarm rate of Random Forest for intrusion detection. The researchers tried with several amounts of trees, from the highest 800 to the lowest 10 layers of trees, to find the best configuration for Random Forest. The outcomes showed that the suggested method, Random Forest with 800 tree layers, obtained accuracy levels of 99.57% on the NSL-KDD dataset, 95.5% on the UNSW-NB15 dataset, and 91.8% on the GPRS dataset. Compared to alternative models like Decision Trees and Multilayer Perceptrons, these outcomes were better. However, if the model has 800 layers, it might take longer to train and test, meaning a more powerful computer would be needed to run it effectively.

[7] Adnan Helmi Azizan, Salama A. Mostafa, and other researchers have also attempted to use machine learning to improve network intrusion detection system performance. NSL-KDD and CIC-IDS2017 were the two main datasets used by these researchers for machine learning model testing and training. Three machine learning models—Decision Jungle, Random Forest, and Support Vector Machine—were assessed in the study papers. The studies used confusion matrices to understand true and false positives and negatives to evaluate the models' performances. To assess the models' performance, performance metrics like accuracy, recall,

and precision were also used along with a 10-fold cross-validation technique. The final findings will only address the Random Forest model because this study focuses on it. The final findings showed that Random Forest has an accuracy of 96.76%, recall of 97.62%, and precision of 97.96%. In this research report, Random Forest's overall performance is average; in contrast, Rifkie Primartha and Bayu Adhi Tama's research performs better.

[8] In a 2016 research paper, Nabila Farnaaz and M. A. Jabbar used the Random Forest Model for IDS. The NSL-KDD dataset was used to train the model. The researchers performed a more thorough performance study with multi-class classification, concentrating specifically on DoS, Probe, R2L, and U2R attacks, as opposed to only binary classification. The accuracy, detection rate, false alarm rate, and Mathew correlation coefficient were then used to assess the outcomes. To further evaluate the model, the researchers used a 10-fold cross-validation technique. The Random Forest with 100 layers of trees obtained an average accuracy of 99.67%, according to the final results. Additionally, the average performance metric for Random Forest using J48 Tree was 99.27%. Following the utilization of FSS-Symmetric Uncertainty, 99.67% accuracy was maintained. Overall, this research report performed noticeably better than the other two investigations.

[9] Deep Neural Networks (DNNs) were trained and tested on the CIC-IDS2017 dataset by Petros Toupas, Dimitra Chamou, Konstantinos M. Giannoutakis, and others. To remove noise from the dataset, the paper started with data cleansing. The dataset was then transformed using the Yeo-Johnson transformation, which was used to normalize the data. Furthermore, approaches of under-sampling and over-sampling were utilized to attain a more equitable distribution of data. An eight-layered Deep Neural Network, consisting of 140, 120, 100, 80, 60, 40, 20, and 120 nodes in each layer, was used by the researchers to train the dataset. Metrics including accuracy, precision, recall, F1 score, and false positive rate were used to assess the final results. With precision at 94.31%, recall at 95.62%, and an F1 score of 94.1%, the multi-class classification's overall accuracy was found to be 99.95% when 10-fold cross-validation splits were used.

[10] For IDS, Li-Hua Li, Ramli Ahmad, Wen-Chung Tsai, and Alok Kumar Sharma suggested utilizing a Deep Neural Network (DNN) based on feature selection. The KDD99 dataset was chosen by the researchers as the main dataset for testing and training their suggested algorithm.

The study examined five different DNN model architectures, ranging from one hidden layer to five hidden layers. Furthermore, the training data go through feature selection in order to compare the performance of the five DNN models with and without feature selection preprocessing. The study included performance criteria, including accuracy, precision, recall, and F1 score, to evaluate the models' efficacy. The DNN model with three hidden layers showed the greatest results with feature selection preprocessing; it achieved a 99.4% accuracy, 99.7% precision, 97.9% recall, and 98.8% F1 score in binary classification for IDS.

[11] Deep Autoencoder was the method used by researchers Srikanthyadav Moraboena, Gayatri Ketepalli, and Padmaja Ragam for IDS. To assess the model, they used the CIC-IDS2017 and NSL-KDD datasets. The study explored various batch sizes, including 32, 64, 128, 256, 512, and 1024, using autoencoder as the methodology. The outcomes were assessed using performance indicators including F1-score, recall, accuracy, and precision. With a final accuracy of 91.76%, precision of 96.27%, and recall of 96.40%, the deep learning model with Autoencoders and a batch size of 1024 proved to be the most successful. Nevertheless, this model's overall accuracy was not as great as that of Deep Neural Networks with numerous hidden layers and Random Forest.

[12] Deep Learning and hyperparameter optimization were also used by Yesi Novaria, Kunang, Siti Nurmaini, Deris Stiawan, and Bhakti Yudho Suprapto for attack classification in IDS. They used two datasets, NSL-KDD and CSE-CIC-IDS2018, to assess the multi-class classification performance of models in their research. They used criteria including accuracy, detection rate, false alarm rate, precision, and F1-score to evaluate the models. To find the optimal model, they employed a deep autoencoder model that was tuned using hyperparameters. Important hyperparameters that were thoroughly examined and refined in the study were the number of hidden layers in deep autoencoders and nodes, learning rate, kernel initialization, batch size, and activation function. The CSE-CIC-IDS2018 dataset had a detection rate of up to 95.79%, whereas the NSL-KDD dataset had a detection rate of 83.33%, according to the final results.

2.1.1 Limitations of Previous Researchers' Models

Previous research on intrusion detection system (IDS) models has identified many limitations that affect model performance. In the work by [6], the emphasis on calculating the number of trees for Random Forest (RF) to obtain higher accuracy outweighed crucial issues such as feature selection and hyperparameter tuning. As a result, their RF-800 model did not achieve high accuracy, demonstrating the need for these optimization processes. Similarly, [7] did not include pre-processing or hyperparameter adjustment in their investigation, resulting in much worse performance across all models examined. Although [8] used Symmetrical Uncertainty for feature selection, the lack of hyperparameter adjustment hindered the model's optimization potential. [9] performed extensive pre-processing but skipped hyperparameter adjustment, potentially affecting precision due to a larger false detection rate. [10] performed extensive feature selection but did not investigate hyperparameter tuning for their Deep Neural Network (DNN) models, hence missing out on potential performance improvements. [11] concentrated on hyperparameter research for batch sizes in the Deep Autoencoder (DAE) model, but lacked thorough pre-processing and feature selection, limiting total model performance. Finally, [12] achieved feature selection and hyperparameter adjustment in their study but found reduced accuracy, potentially due to architectural concerns such as the number of hidden layers in the DAE model. Addressing these limitations with complete optimization methodologies that include feature selection, hyperparameter tuning, and data pre-processing should considerably improve the performance and resilience of IDS models in future research attempts.

2.2 Previous Researchers' Works on Data Pre-processing and Feature Selections

Two notable research publications provide useful insights into data preprocessing and feature selection strategies. In the study [13], Ebrima Jaw and Xueming Wang emphasize the importance of data pre-processing in improving data quality for optimal model training results. Their methodology, which included data cleaning, label encoding, normalization, clustering, and a multi-step feature selection procedure, resulted in a set of 13 features suited specifically for CIC-IDS2017. This thorough technique resulted in a remarkable accuracy increase of up to 99.99%. Despite the complexity of their feature selection method, the significant accuracy improvement justifies the lengthy procedure.

In contrast, Ishita Karna and others [14] used a simple yet effective technique to improve dataset quality. Their ensemble-based filter feature selection technique combined numerous datasets into a consistent format, handled missing values, normalized data, and used mutual information, Chi-Squared scores, and Pearson's correlation coefficient to choose features. This method resulted in 24 and 25 features for the majority and minority datasets, respectively, from the original 77 features in CIC-IDS 2017. Despite its simplicity, this method performed similarly to the more complex approach used in [13], making it an appealing option due to its ease of implementation without sacrificing accuracy.

The comparison between these two approaches demonstrates the flexibility and efficacy of various data pre-processing and feature selection strategies. While a more complex method may result in significant accuracy gains, a simplified strategy can produce similar outcomes with more simplicity and ease of implementation. These findings will help to choose the best approach for this research article, with the goal of striking a balance between accuracy enhancement and practicality in feature selection approaches.

2.3 Previous Researchers' Works on Hyperparameter Tuning

Hyperparameter tuning is critical in optimizing machine learning models for various problem domains, as stressed by Li Yang and Abdallah Shami in [15]. Understanding the optimal hyperparameter method is critical for improving the performance of machine learning models in intrusion detection systems. There are several approaches for hyperparameter optimization, including model-free algorithms, gradient-based optimization, and Bayesian optimization. Grid search, a popular strategy, methodically investigates hyperparameters in a model using a brute-force method, making it especially successful for categorical hyperparameters and simple to implement. However, its main disadvantage is that it is time-consuming, with a time complexity of $O(N^k)$ as the search space expands.

To solve the limitations of grid search, researchers have investigated alternate methods such as random search. Random search tests values in the hyperparameter search space by randomly picking samples, allowing for parallelization due to the independence of each evaluation. The strategy successfully decreased the time complexity to O(N). Despite its benefits, random search brings new limits by not taking into consideration past findings, perhaps resulting in the loss of key insights gained during the optimization process.

For example, in [12], researchers used the Talos library, which is known for its automated hyperparameter optimization capabilities, to tune parameters. The study used binary and categorical cross-entropy on the dataset and model to get discrete features only, speeding the hyperparameter optimization procedure. While this automated approach facilitates in the identification of significant hyperparameters and the efficient exploration of their combinations, it incurs processing costs due to the evaluation of multiple models to establish the ideal hyperparameter composition.

In conclusion, investigating alternative hyperparameter optimization approaches is critical for improving the performance of machine learning models in IDS. While grid search and random search have different advantages and disadvantages, automated approaches, such as those employing the Talos library, accelerate the process by focusing on important hyperparameters, although at a higher computational cost. Balancing the advantages and disadvantages between efficiency and computational cost is critical in determining the best hyperparameter tuning technique for IDS models.

2.4 Summary

A literature assessment of machine learning and deep learning models for intrusion detection systems (IDS), such as Random Forest, Deep Neural Network (DNN), and Deep Autoencoders (DAE), revealed some critical conclusions. The review emphasized the models' performance in multiple investigations, concentrating on criteria such as accuracy, precision, recall, and F1 score.

The study found that, while multiple studies obtained accuracy rates of 90% or greater, many limitations were identified. One frequent drawback was the absence of hyperparameter tweaking in the models, resulting in less valuable accuracy and precision. Notably, experiments that used hyperparameter tuning, such as those using automated optimization procedures like the Talos package, demonstrated better performance.

Furthermore, the review looked into data pre-processing and feature selection techniques. Various ways were seen, ranging from complex feature selection processes using ensemble systems to simplified methods using popular techniques such as mutual information, Chi-Squared score, and Pearson's correlation coefficient.

Moving forward, the study intends to solve these challenges by emphasizing pre-processing with feature selection and comprehensive hyperparameter optimization. The purpose of an optimized pre-processing strategy, combined with effective feature selection approaches and automated hyperparameter optimization, is to improve model performance in terms of accuracy, precision, recall, and F1 score for IDS applications.

Chapter 3

System Methodology/Approach

3.1 Research Methodology

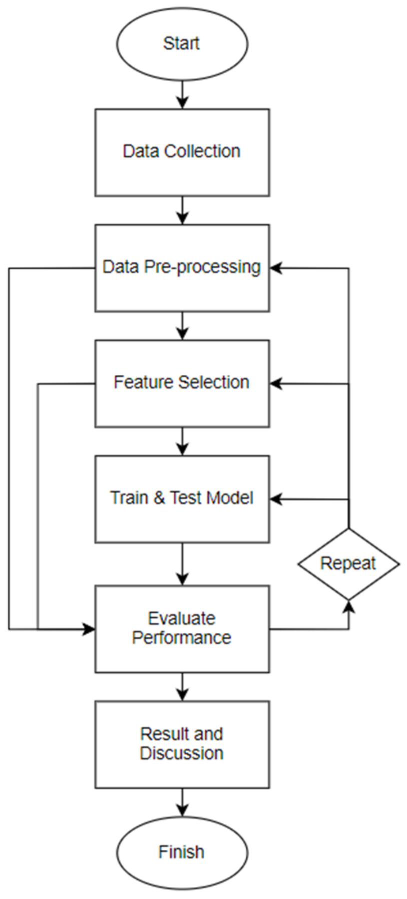


Figure 3.1.1 Research Methodology

The project will begin with data collection, utilizing the CIC-IDS 2017 dataset as previously mentioned. Subsequently, this data will be imported into Jupyter Notebook for further processing. Python libraries will be employed for data importation and pre-processing. During the pre-processing of the IDS data, a feature selection process will be implemented to extract only the essential attributes.

Following this, the data will be divided into training and testing sets to facilitate model training and result prediction. These outcomes will be assessed using performance metrics such as accuracy, precision, recall, and the confusion matrix to gain insights into the overall model performance. Moreover, some performance metrics will be applied during the pre-processing and feature selection stages to ensure effective data processing, leading to improved outcomes in future model training.

The stages of training, testing, and performance evaluation will be iterated multiple times to obtain performance results for three machine learning models. Subsequently, these results will be compared and contrasted with previous researchers' results after applying hyperparameter tuning to the models. The findings and ensuing discussion will be meticulously documented in the project report.

3.2 Dataset Requirement – CIC-IDS2017

In the project, the machine learning model was trained and assessed using the CIC-IDS 2017 dataset. This dataset mimics human activity patterns and serves as an intrusion detection dataset. It handles a variety of network traffic, including both regular and malicious attacks, including DoS, Brute Force, and Web attacks. Before being used to train and test the machine learning models, this large dataset will go through pre-processing and feature selection.

3.3 Data Pre-processing and Feature Selection

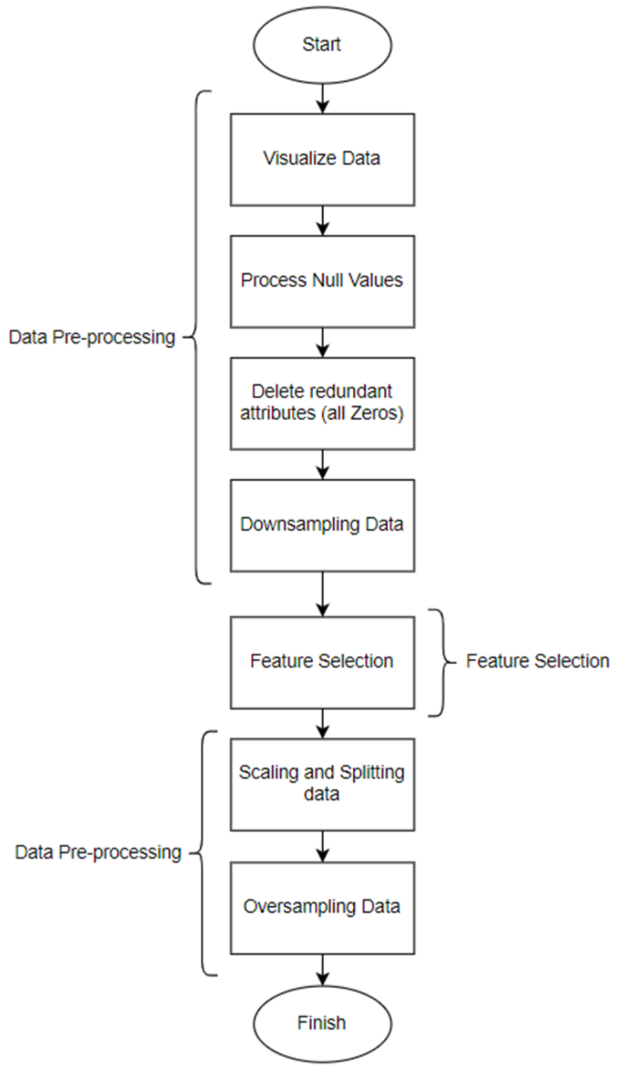


Figure 3.3.1 Data Pre-processing and Feature Selection

The initial key component of this project involves data pre-processing, a crucial phase aimed at ensuring the data is appropriately prepared and refined for subsequent training and testing. The dataset will be visualized in the first stage to understand the data structure and data type. Subsequently, any identified null values within the data will be addressed through deletion. This decision is driven by the substantial size of the dataset, which remains sufficient for future data training while having a minimal impact. Besides that, the attributes with zero values had been identified and deleted as these attributes are redundant and might cause confusion in model prediction. Following this, the data will undergo down sampling to ensure an unbiased and balanced representation of normal and attacked traffic data.

The data pre-processing process will then be followed by feature selection. Feature selection aims to identify the most significant features within the dataset, enhancing prediction accuracy by reducing the presence of confounding attributes. In this research, Pearson's Correlation is employed as the feature selection technique as it is the most frequently used method for continuous data.

Subsequently, the dataset undergoes additional pre-processing through scaling using the Quantile Transformer Scaler. This scaler transforms the variable distribution of data into a normal distribution, ranging from zero to one [16]. Scaling the data ensures that the model can more effectively and consistently learn from the data. Finally, oversampling is applied to the data to establish a more balanced and unbiased representation of each category of attacked traffic.

3.4 Modelling and Hyperparameter Tuning

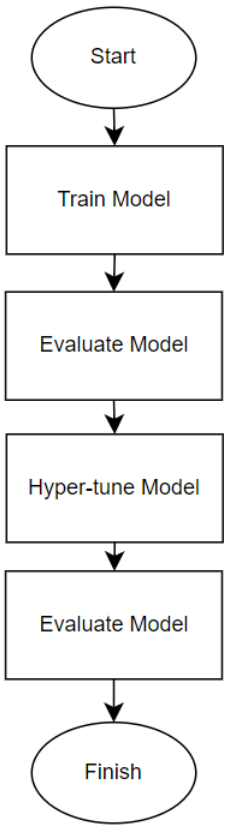


Figure 3.4.1 Training, Testing and Hyperparameter Tuning Model

After going through the process of pre-processing the data and splitting the data into training and testing sets, the data is now ready for the model to train. The research will first train the model with the original parameter to evaluate the original model's performance. Then, the model will go through hyperparameter tuning to find out the optimum model for the algorithm. The optimum model will then be evaluated again using the four performance metrics, accuracy, precision, recall, and F1 score. The percentage obtained from both original and hyperparameter-tuned models will be recorded in a table to compare with the previous researchers' results.

3.4.1 Modelling

The research paper will explore three models that will be implemented in the research. These models are Random Forest, Deep Neural Network, and Deep Autoencoder.

3.4.1.1 Random Forest

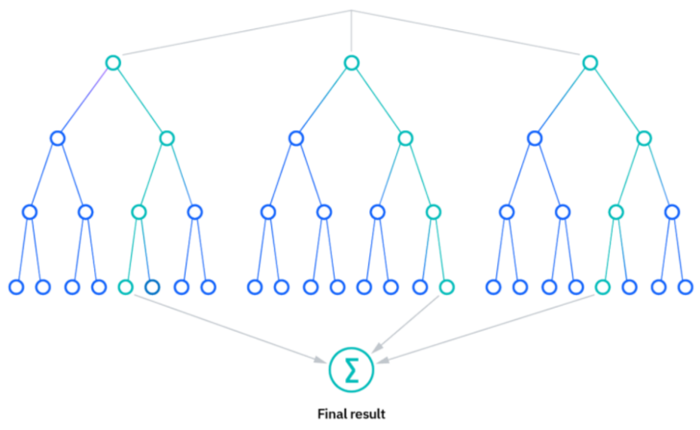


Figure 3.4.1.1 The example of RF [17]

Leo Breiman and Adele Cutler introduced the Random Forest algorithm as a type of machine learning [17]. With several nodes and branches, it performs as a forest of decision trees. By offering a more thorough classification of labelled data, the Random Forest algorithm improves the accuracy of the model's output [17]. The size of the nodes, the number of trees, and the number of features sampled are the three main hyperparameters in the Random Forest classifier [17]. With a lower chance of overfitting the data, this approach reduces risk and enables flexibility in handling both regression and classification problems [17].

3.4.1.2 Deep Neural Network

Deep Neural Network (DNN) is a deep learning technique that is commonly employed in data-driven modelling [18]. It is a stacked neural network, which predicts output variables based on input variables by connecting multiple layers with multiple nodes and edges [18] [19]. DNN is a typical approach for calculating the probability of each class and differentiating between valid and malicious traffic since it helps manage unstructured and unlabelled data [20].

3.4.1.3 Deep Autoencoder

The Deep Autoencoder algorithm, first presented by Vincent et al. in 2008 [21], is an improvement on the classic Autoencoder. The Autoencoder is a feedforward neural network with one hidden layer that operates similarly to the Multi-layer Perceptron algorithm. The process consists of two phases: the first is unsupervised learning to acquire features, and the second is supervised learning to optimize the network [21]. Deep Autoencoder can improve the resilience and denoise training data in comparison to regular Autoencoder [21].

3.5 Evaluation Methods

The research paper will use mainly four performance metrics to evaluate the performance of the models stated above. The four metrics are accuracy, precision, recall, and F1 score.

3.5.1 Confusion Matrix

A Confusion Matrix is a N x N matrix that is frequently used to evaluate classification model performance [22]. By comparing actual target values with predicted values, it offers a thorough understanding of the model and yields important metrics like True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Table 3.1 The Confusion Matrix

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

3.5.2 Accuracy

Accuracy is one of the most widely used criteria for assessing classification model performance [23]. It enables an understanding of the model's accuracy percentage in determining the amount of network traffic attacks. The following equation shows how to determine a model's accuracy:

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

3.5.3 Precision

Precision is another key parameter used in classification models [23]. It helps in understanding the probability that attacks detected by the model are actual attacks rather than false alarms. Administrators could benefit from a lower false alarm rate, which is ensured by increasing precision. The following equation shows how to determine a model's precision:

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

3.5.4 Recall

Regardless of the false alarm rate, recall is a critical parameter that guarantees the identification of all threats. It is especially helpful when False Positives are less concerning than False Negatives [23]. The following equation illustrates how to determine a model's recall:

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

3.5.5 F1 Score

The F1 score provides a more balanced average result by combining the Precision and Recall results [23]. The following equation shows how to determine a model's F1 score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.6 Software Requirement – Jupyter Notebook and Python

Jupyter Notebook is software that offers a code-writing notebook format, making data exploration and visualization simple [24]. Python is the most widely used language in deep learning and machine learning, and the software supports the language. For this reason, Jupyter Notebook will be used in this research writing to explore data and visualize code snippets.

3.7 Summary

The research methodology described in Chapter 3 is a thorough approach to constructing and assessing machine learning models for intrusion detection systems (IDS) based on the CIC-IDS 2017 dataset. The chapter begins with data collection and imports into Jupyter Notebook, followed by data preparation and feature selection to extract significance attributes. Afterwards, the data is divided into training and testing sets to train the model and evaluate its performance using metrics like accuracy, precision, recall, and the confusion matrix.

The approach goes through several training, testing, and evaluation cycles for three machine learning models: Random Forest, Deep Neural Network (DNN), and Deep Autoencoder (DAE). Hyperparameter tuning is used to optimize model performance, and the results are compared with previous researchers' findings. The chapter also discusses dataset requirements, data pre-treatment methods such as handling null values, down sampling, feature selection with Pearson's Correlation, and scaling strategies.

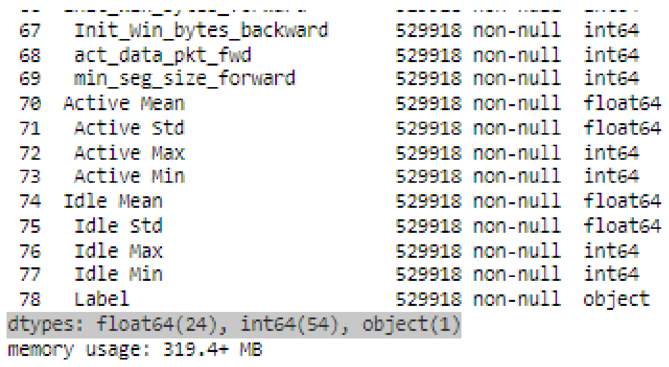
The training, testing, and hyperparameter tuning models are described in detail, with a focus on the specifics of each model's implementation and the benefits it offers IDS applications. Evaluation approaches, such as the confusion matrix, accuracy, precision, recall, and F1 score, are thoroughly examined, giving a comprehensive framework for measuring model performance.

Finally, Jupyter Notebook and Python outlined as the software requirements for the study, emphasizing the significance of the tools required in this study for data exploration, code visualization, and model creation.

Chapter 4

Data Pre-Processing

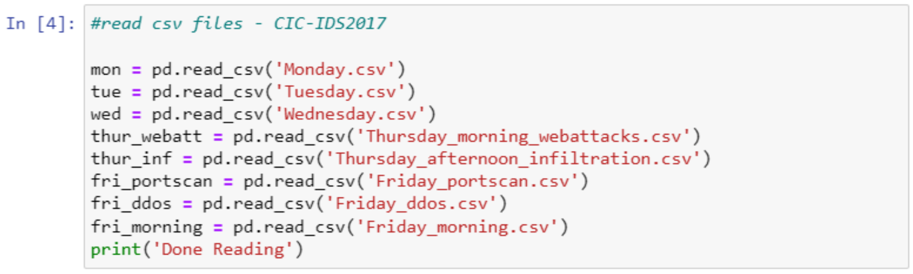
4.1 Data Pre-processing and Feature Selection Work



```
 67   Init_Win_bytes_backward      529918 non-null  int64
 68   act_data_pkt_fwd             529918 non-null  int64
 69   min_seg_size_forward         529918 non-null  int64
 70  Active Mean                   529918 non-null  float64
 71   Active Std                   529918 non-null  float64
 72   Active Max                   529918 non-null  int64
 73   Active Min                   529918 non-null  int64
 74  Idle Mean                     529918 non-null  float64
 75   Idle Std                     529918 non-null  float64
 76   Idle Max                     529918 non-null  int64
 77   Idle Min                     529918 non-null  int64
 78   Label                        529918 non-null  object
dtypes: float64(24), int64(54), object(1)
memory usage: 319.4+ MB
```

Figure 4.1 The information of the dataset for Monday.csv

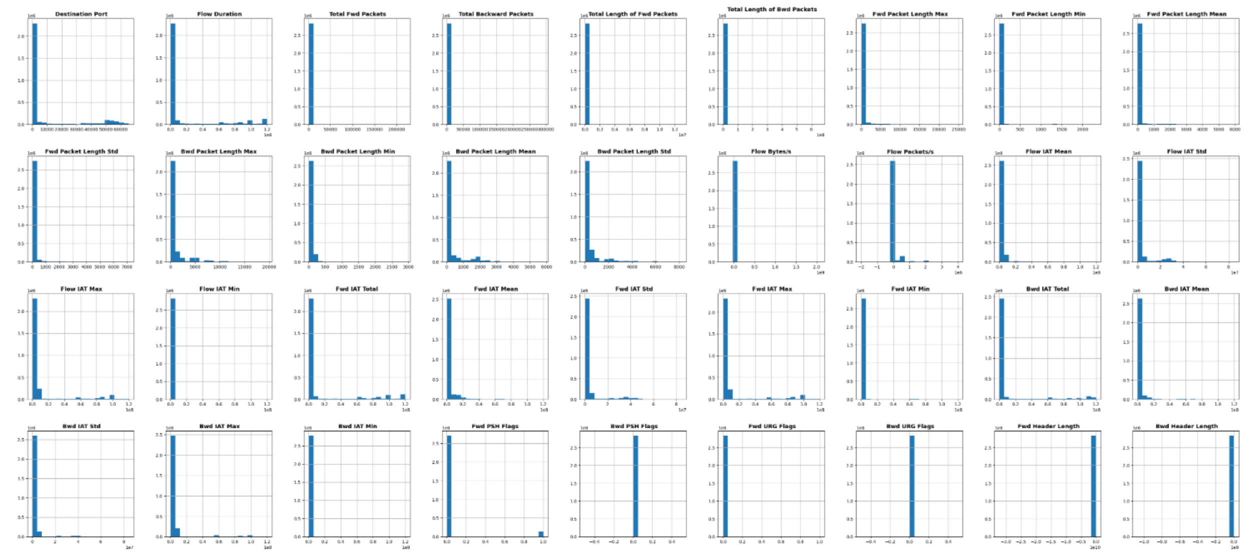
The pre-processing step starts by reading the eight csv files of dataset and visualizing the data. The CIC-IDS2017 dataset consists of total 79 columns and estimated of total 3,000,000 of rows, including all csv files. The attributes consist of mainly continuous data, with 24 attributes in float type and 54 attributes in integer type. The dataset only consisting of one object which is "Label", where it labels the sample as "Benign" for normal traffic and attacked traffic with attack type such as "FTP-Patator", "SSH-Patator", "DoS slowloris" and many more. Afterwards, the data had been checked if there are any null values in each attribute through "is null" function. The result shows that all the csv files contained null values on Flow Bytes and Flow Packets. The rows with null values will be dropped as the total data in this dataset is enough to be trained by machine learning model. The purpose of dropping null values is to ensure that the dataset only consisting of the valid data, avoiding model to get confused by the null values.



```
In [4]: #read csv files - CIC-IDS2017

        mon = pd.read_csv('Monday.csv')
        tue = pd.read_csv('Tuesday.csv')
        wed = pd.read_csv('Wednesday.csv')
        thur_webatt = pd.read_csv('Thursday_morning_webattacks.csv')
        thur_inf = pd.read_csv('Thursday_afternoon_infiltration.csv')
        fri_portscan = pd.read_csv('Friday_portscan.csv')
        fri_ddos = pd.read_csv('Friday_ddos.csv')
        fri_morning = pd.read_csv('Friday_morning.csv')
        print('Done Reading')
```

Figure 4.2 The csv files read

The pre-processing continued by combining the eight CSV files into one dataset. Each CSV file has a different type of traffic, for instance, “Monday.csv” consists of only normal traffic while “Thursday_morning_webattackcks.csv” consists of a combination of normal traffic and traffic that suffers on web attack, including brute force, XSS, and SQL injection. As shown in the figure above, the files consisting of the attacked traffic had been named on the CSV files beforehand for easy handling. By combining these data files, the model can train with a bigger set of data, increasing the performance of the model. Besides, the model is going to train through binary classification and multi-output classification. Therefore, it is necessary to combine them to process them in the future. After combining the data, the total row in the dataset is 2,827,876.



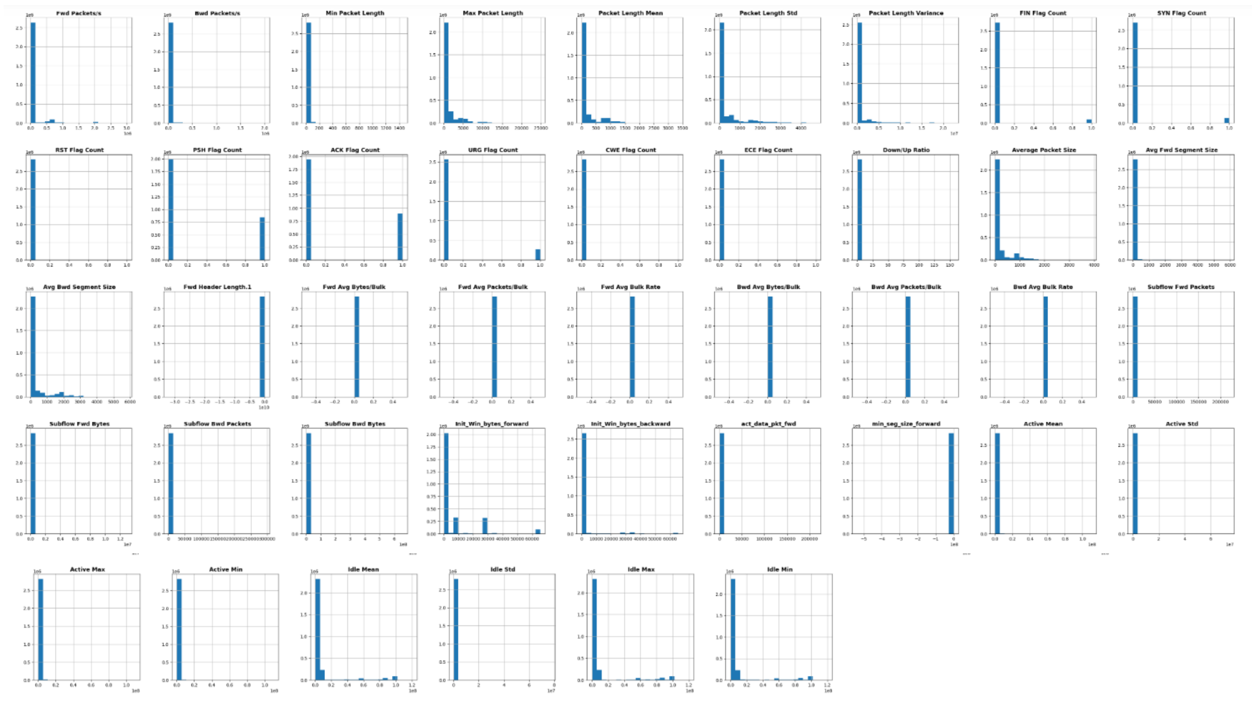
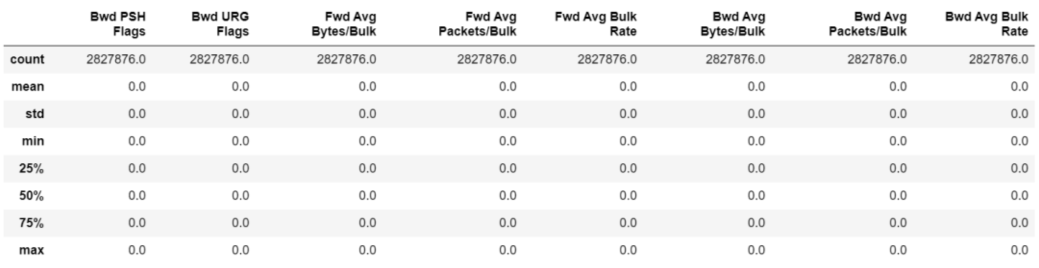


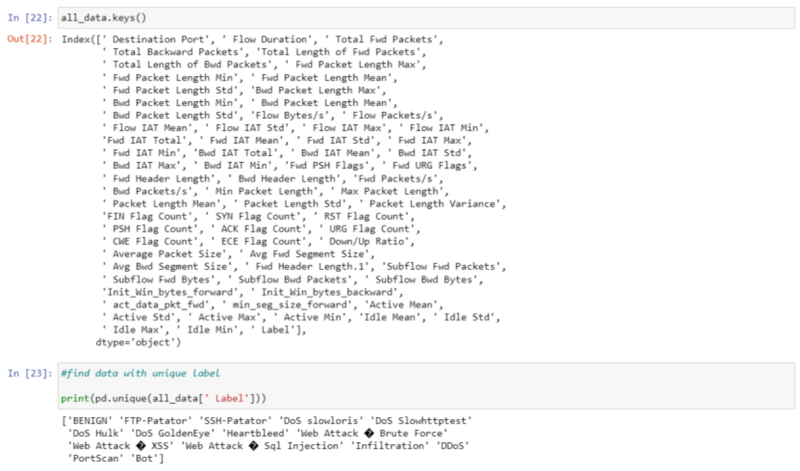
Figure 4.3 Histogram on Features' Distribution



	Bwd PSH Flags	Bwd URG Flags	Fwd Avg Bytes/Bulk	Fwd Avg Packets/Bulk	Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk	Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate
count	2827876.0	2827876.0	2827876.0	2827876.0	2827876.0	2827876.0	2827876.0	2827876.0
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 4.4 Attributes with all zeroes value

After dropping the null values, the histograms on each attribute were generated to view the distribution of each attribute's values. As shown in the above figure, most of the histograms have zero values. The statement is supported by the table generated afterward, where there are a total of eight attributes that have zero values. These attributes are redundant as they did not give any characteristics on defining the packets are either normal or attacked. Thus, the attributes are dropped from the dataset to prevent the model from training unnecessary features.



```
In [22]: all_data.keys()

Out[22]: Index([' Destination Port', ' Flow Duration', ' Total Fwd Packets',
                ' Total Backward Packets', 'Total Length of Fwd Packets',
                ' Total Length of Bwd Packets', ' Fwd Packet Length Max',
                ' Fwd Packet Length Min', ' Fwd Packet Length Mean',
                ' Fwd Packet Length Std', 'Bwd Packet Length Max',
                ' Bwd Packet Length Min', ' Bwd Packet Length Mean',
                ' Bwd Packet Length Std', 'Flow Bytes/s', ' Flow Packets/s',
                ' Flow IAT Mean', ' Flow IAT Std', ' Flow IAT Max', ' Flow IAT Min',
                'Fwd IAT Total', ' Fwd IAT Mean', ' Fwd IAT Std', ' Fwd IAT Max',
                ' Fwd IAT Min', 'Bwd IAT Total', ' Bwd IAT Mean', ' Bwd IAT Std',
                ' Bwd IAT Max', ' Bwd IAT Min', 'Fwd PSH Flags', ' Fwd URG Flags',
                ' Fwd Header Length', ' Bwd Header Length', 'Fwd Packets/s',
                ' Bwd Packets/s', ' Min Packet Length', ' Max Packet Length',
                ' Packet Length Mean', ' Packet Length Std', ' Packet Length Variance',
                'FIN Flag Count', ' SYN Flag Count', ' RST Flag Count',
                ' PSH Flag Count', ' ACK Flag Count', ' URG Flag Count',
                ' CWE Flag Count', ' ECE Flag Count', ' Down/Up Ratio',
                ' Average Packet Size', ' Avg Fwd Segment Size',
                ' Avg Bwd Segment Size', ' Fwd Header Length.1', 'Subflow Fwd Packets',
                ' Subflow Fwd Bytes', ' Subflow Bwd Packets', ' Subflow Bwd Bytes',
                'Init_Win_bytes_forward', ' Init_Win_bytes_backward',
                ' act_data_pkt_fwd', ' min_seg_size_forward', 'Active Mean',
                ' Active Std', ' Active Max', ' Active Min', 'Idle Mean', ' Idle Std',
                ' Idle Max', ' Idle Min', ' Label'],
               dtype='object')

In [23]: #find data with unique label

         print(pd.unique(all_data[' Label']))

         ['BENIGN' 'FTP-Patator' 'SSH-Patator' 'DoS slowloris' 'DoS Slowhttptest'
          'DoS Hulk' 'DoS GoldenEye' 'Heartbleed' 'Web Attack � Brute Force'
          'Web Attack � XSS' 'Web Attack � Sql Injection' 'Infiltration' 'DDoS'
          'PortScan' 'Bot']
```

Figure 4.5 Unique Labels and Attributes

The dataset lacked proper naming for each attribute and attack name. For instance, the attributes contained unnecessary spaces in front of their names. Furthermore, the labels for the samples had unique and unreadable characters, as shown in the figure. Therefore, these labels have been renamed with readable labels, while each attribute with unnecessary spacing has had the spacing removed. These steps enable better processing for research with more consistent naming and easier readability of the labels for each row.

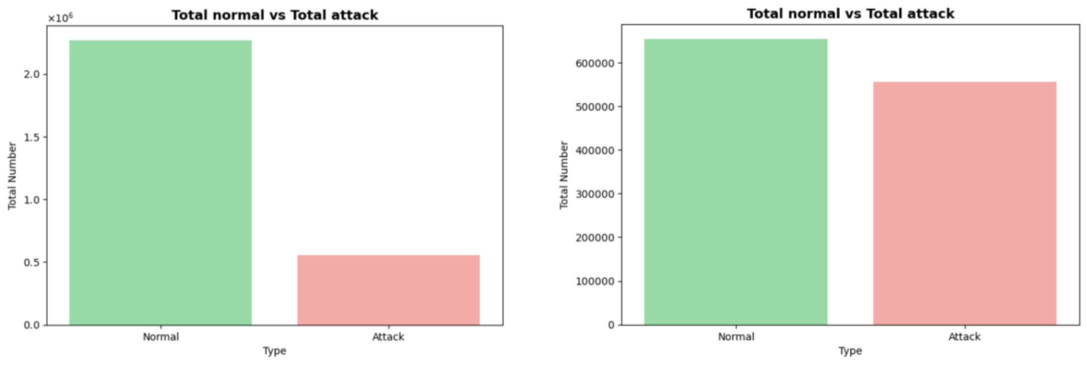


Figure 4.6 Total normal vs Total attack (Before and After)

The histogram of total normal versus total attack was generated to understand the ratio of normal traffic and attacked traffic in the dataset. As the dataset had been combined using a total of 8 CSV files from Monday to Friday, it consists of lots of data that are labelled as normal. For instance, the Monday CSV file consists of only normal traffic to understand the distribution of normal traffic. The imbalanced dataset might cause the model training to be highly biased.

Thus, under-sampling was done by using a Random Under Sampler. The dataset had been under-sampled from 2,827,876 samples to 1,211,327 samples.

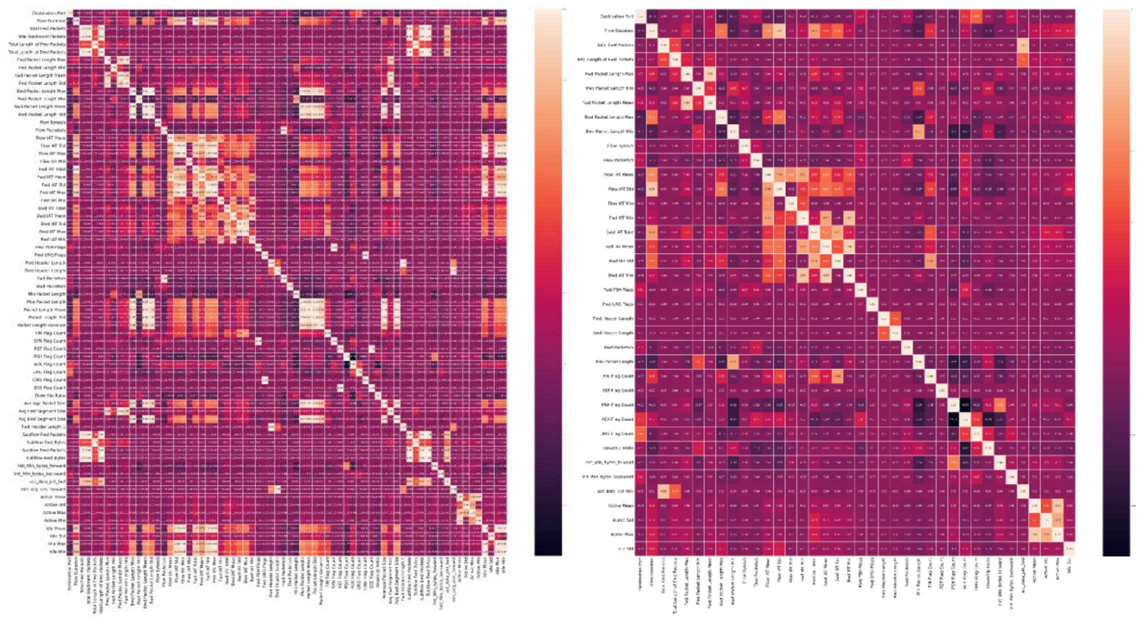
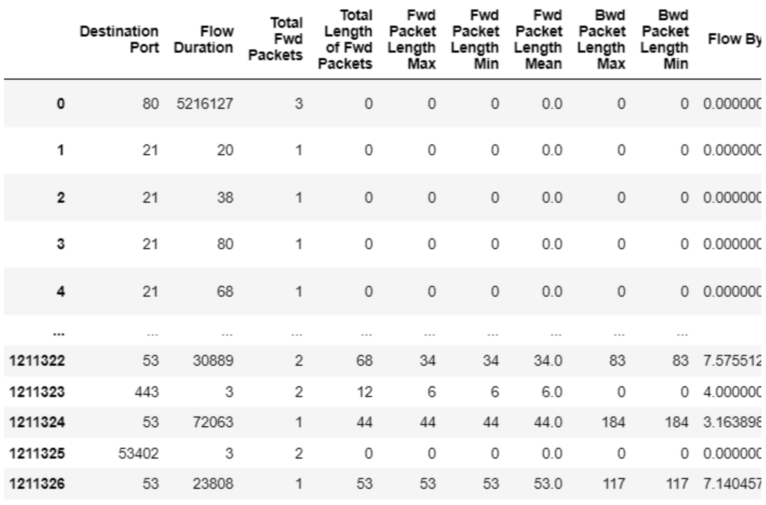


Figure 4.7 Correlation matrix before and after feature selection

The pre-processing step continued with the feature selection, where Pearson's Correlation was used to find out the correlation of each feature. As shown in the figure, the correlation matrix before the feature selection consists of a few attributes that are least significant for the dataset. For instance, idle max and idle min have a high correlation with a few attributes such as Flow IAT, Flow Duration, and more, which range from 0.83 to 1.00. Therefore, this indicates that the idle min and idle max are giving almost the same result as the dependent variables and viewed as redundant features in this case. Therefore, these insignificant features dropped from 72 features to 40 features.



	Destination Port	Flow Duration	Total Fwd Packets	Total Length of Fwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Bwd Packet Length Max	Bwd Packet Length Min	Flow By
0	80	5216127	3	0	0	0	0.0	0	0	0.000000
1	21	20	1	0	0	0	0.0	0	0	0.000000
2	21	38	1	0	0	0	0.0	0	0	0.000000
3	21	80	1	0	0	0	0.0	0	0	0.000000
4	21	68	1	0	0	0	0.0	0	0	0.000000
...	...	...	...	...	...	...	...	...	...	
1211322	53	30889	2	68	34	34	34.0	83	83	7.575512
1211323	443	3	2	12	6	6	6.0	0	0	4.000000
1211324	53	72063	1	44	44	44	44.0	184	184	3.163898
1211325	53402	3	2	0	0	0	0.0	0	0	0.000000
1211326	53	23808	1	53	53	53	53.0	117	117	7.140457

Figure 4.8 The distribution of values in each attribute

As shown in the figure, the attributes have various values and vary range. This might cause one of the attributes with a big value to dominate the model, causing the model's performance to be low. Therefore, these attributes' values will be scaled to a similar scale, ensuring they contribute equally to the model instead of one of them dominating the model. As most data in the dataset is numerical and has a huge variance with each other, thus, these data are scaled using the Quantile Transformer to transform the data distribution into normal distribution by dropping the label of each data. Scaling the data helps to balance the features' impact on the model training and improve the performance of algorithms.

Then, the data was split in the ratio of 70 and 30 to the training set and test set. Not only that, as the research will train the model using both binary classification and multi-output classification, the y_train and y_test are further mapped as both normal and attack traffic, without the detailed naming of attacks for binary classification purposes.

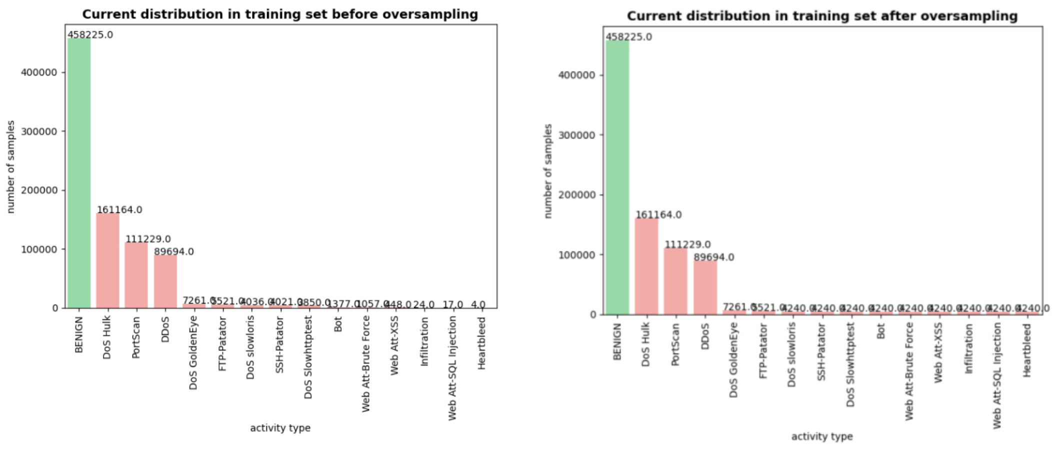


Figure 4.9 Distribution of training set before and after oversampling

In the previous step, the data was down sampled to ensure that the normal traffic and attacked traffic were balanced. However, each attacked traffic category is imbalanced as DoS Hulk has the highest amount of sample while Heartbleed has the least. It will cause the model to perform poorly in multi-output classification as the samples for each attacked traffic category are not similar and might cause biased results. Thus, oversampling is done by using Smote to balance and increase the sample size of attacked traffic with a low amount of sample. To ensure that the important patterns will not leak into the test set, the oversampling will only done on a training set of data. As a result, the categories with a low sample size had been oversampled to 4240 samples while other categories remained the same.

The labels in the dataset are in object type, categorizing entries as either "benign" for normal traffic or various categories of attacked traffic. To facilitate model testing and performance prediction, it's essential to encode these categorical values into binary format. Two encoding methods, which are One Hot Encoder and Label Encoder, were utilized in this research. The binary classification label set employed the Label Encoder to transform normal traffic and attacked traffic into 0s and 1s. Conversely, the multi-output classification label set used the One Hot Encoder to convert them into binary arrays with 14 columns.

Finally, the training and testing sets are saved as CSV files for future use. These CSV files encompass the raw training and testing sets, up-sampled training and testing sets, binary classification training and testing sets, and multi-output classification training and testing sets. Storing these various combinations of data will enable the models to be tested with different configurations, ensuring optimal performance, and facilitating meaningful comparisons in future research. The total time usage for the data pre-processing and feature selection is 10 minutes.

4.2 Summary

The chapter examines the key components of data pre-processing and feature selection in the context of developing an intrusion detection system (IDS) model with the CIC-IDS2017 dataset. Pre-processing stages involve dealing with null values, joining CSV files, removing redundant features, renaming attributes and labels, addressing data imbalance, and scaling attributes to ensure balanced model training and optimal performance.

Initially, the dataset's structure and properties are described, including the presence of null values and redundant features. The pre-processing procedures include removing null values and unnecessary features, integrating CSV files to build a comprehensive dataset, and renaming attributes and labels to improve consistency and readability.

To resolve the imbalance between normal and attacked traffic data, under and over-sampling approaches are used, resulting in a balanced representation of traffic categories for model training. Pearson's Correlation feature selection assists in the identification and removal of redundant features, as well as the optimization of the dataset for model training and dimensional reduction.

Furthermore, utilizing the Quantile Transformer to scale features guarantees that they contribute evenly to the model training process, preventing large-valued attributes from dominating. Encoding categorical labels into binary format is also covered to make binary and multi-output classification jobs easier.

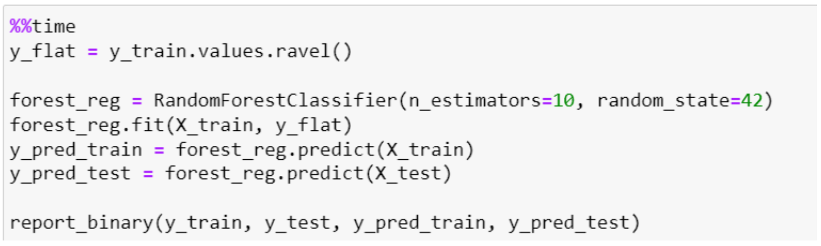
The complete pre-processing and feature selection pipeline improves the dataset's quality and suitability for model training. The chapter concludes with the storage of pre-processed datasets in CSV format for future model testing and evaluation, which takes approximately 10 minutes.

Chapter 5

Modelling and Hyperparameter Tuning

5.1 Random Forest Model

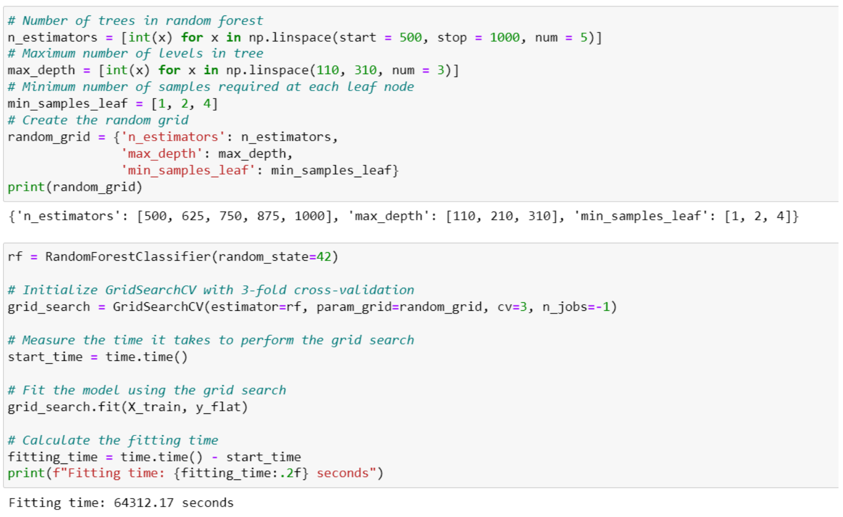
The pre-processed data was sent to a new Jupyter Notebook file for training and testing. The research will first try the model on binary classification, followed by multi-class classification.



```
%%time
y_flat = y_train.values.ravel()

forest_reg = RandomForestClassifier(n_estimators=10, random_state=42)
forest_reg.fit(X_train, y_flat)
y_pred_train = forest_reg.predict(X_train)
y_pred_test = forest_reg.predict(X_test)

report_binary(y_train, y_test, y_pred_train, y_pred_test)
```

Figure 5.1 RF model implementation

The above is the code used for the Random Forest classifier. The model used ten estimators with a random state of 42 initially. Then. The model was fit with the X_train and y_train that were imported from the pre-processed dataset. It was then going through prediction to get the report.



```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 500, stop = 1000, num = 5)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(110, 310, num = 3)]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_leaf': min_samples_leaf}
print(random_grid)
```

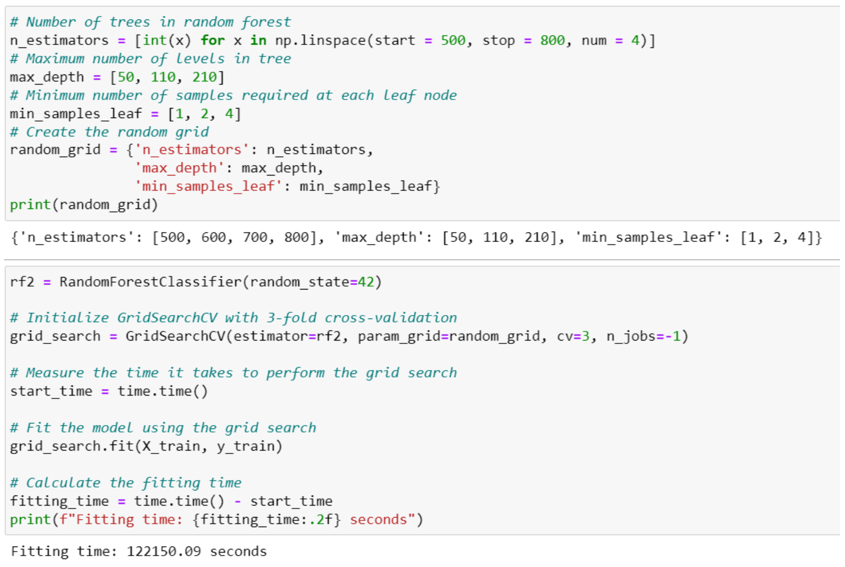
```
{'n_estimators': [500, 625, 750, 875, 1000], 'max_depth': [110, 210, 310], 'min_samples_leaf': [1, 2, 4]}
```

```
rf = RandomForestClassifier(random_state=42)

# Initialize GridSearchCV with 3-fold cross-validation
grid_search = GridSearchCV(estimator=rf, param_grid=random_grid, cv=3, n_jobs=-1)

# Measure the time it takes to perform the grid search
start_time = time.time()

# Fit the model using the grid search
grid_search.fit(X_train, y_flat)

# Calculate the fitting time
fitting_time = time.time() - start_time
print(f"Fitting time: {fitting_time:.2f} seconds")
```

```
Fitting time: 64312.17 seconds
```

Figure 5.2 Hyperparameter tuning on RF model for binary classification

The research then did hyperparameter tuning on the Random Forest Model. The hyperparameters used in this research are estimators ranging from 500 to 1000, maximum depth ranging from 110 to 310, and minimum sample leaves of one, two, or four. To find out the best model from these hyperparameters, the research used 3-fold cross-validation as it provides a more accurate result in a shorter period, compared to 10-fold cross-validation. The total training used was 17.86 hours. The most optimum parameters for this hyperparameter tuning are 625 numbers of estimators, 110 depths, and two sample leaves.



```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 500, stop = 800, num = 4)]
# Maximum number of levels in tree
max_depth = [50, 110, 210]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_leaf': min_samples_leaf}
print(random_grid)
```

```
{'n_estimators': [500, 600, 700, 800], 'max_depth': [50, 110, 210], 'min_samples_leaf': [1, 2, 4]}
```

```
rf2 = RandomForestClassifier(random_state=42)

# Initialize GridSearchCV with 3-fold cross-validation
grid_search = GridSearchCV(estimator=rf2, param_grid=random_grid, cv=3, n_jobs=-1)

# Measure the time it takes to perform the grid search
start_time = time.time()

# Fit the model using the grid search
grid_search.fit(X_train, y_train)

# Calculate the fitting time
fitting_time = time.time() - start_time
print(f"Fitting time: {fitting_time:.2f} seconds")
```

```
Fitting time: 122150.09 seconds
```

Figure 5.3 Hyperparameter tuning on RF model for multiclass classification

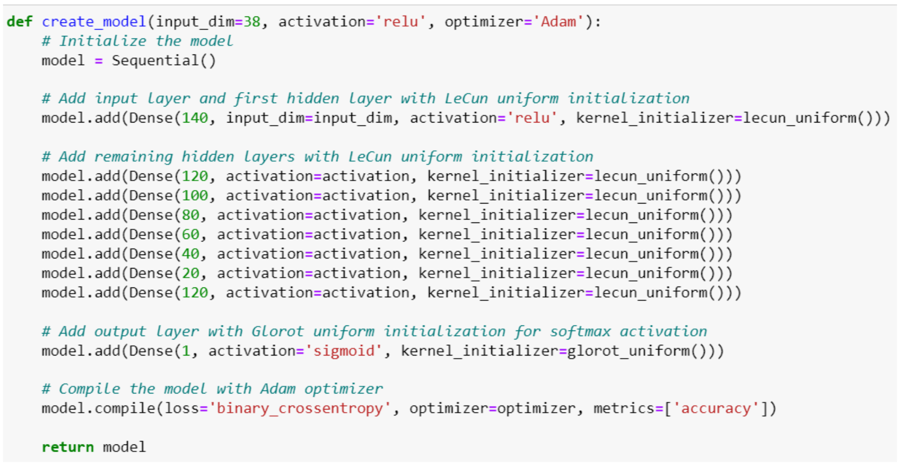
A similar Random Forest model with ten numbers of estimators and 42 random states was used to fit the X_train and y_train for multi-class classification. Then, hyperparameter tuning was applied to the model. The hyperparameter used for this model is number estimators ranging from 500 to 800, maximum depth ranging from 50 to 210, and minimum sample leaves of one, two, and four. After applying it with 3-fold cross-validation, the total grid search time is 33.93 hours. The time used is longer than binary classification because the results are more detailed compared to binary classification. The most optimum parameters for this hyperparameter tuning are 800 numbers of estimators, 110 depths, and 1 sample leaf.

Below are the most optimum Random Forest models for both binary and multi-class classification after hyperparameter tuning:

Table 5.1 The most optimum RF models

Classification Type	Number of Estimators	Maximum Depth	Minimum Sample	Fitting Time (hour)
Binary Class	625	110	2	17.86
Multi Class	800	110	1	33.93

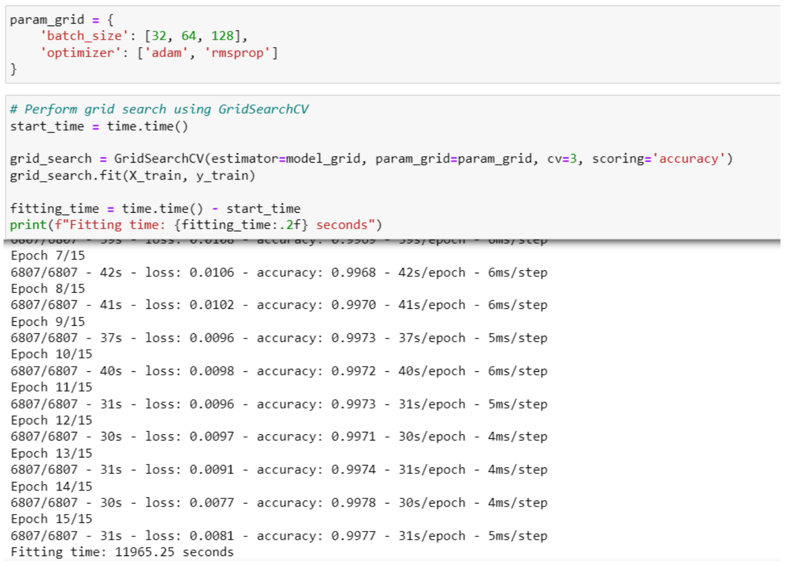
5.2 Deep Neural Network Model



```
def create_model(input_dim=38, activation='relu', optimizer='Adam'):
    # Initialize the model
    model = Sequential()

    # Add input layer and first hidden layer with LeCun uniform initialization
    model.add(Dense(140, input_dim=input_dim, activation='relu', kernel_initializer=lecun_uniform()))

    # Add remaining hidden layers with LeCun uniform initialization
    model.add(Dense(120, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(100, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(80, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(60, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(40, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(20, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(120, activation=activation, kernel_initializer=lecun_uniform()))

    # Add output layer with Glorot uniform initialization for softmax activation
    model.add(Dense(1, activation='sigmoid', kernel_initializer=glorot_uniform()))

    # Compile the model with Adam optimizer
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

    return model
```

Figure 5.4 DNN model implementation for binary classification

The second model used is the Deep Neural Network model (DNN), which is one of the commonly used Deep Learning Models. The model used input layers with 38 dimensions with a total of eight hidden layers. Each layer progressively decreases the number of neurons from 140 to 20, using the ReLu activation function for non-linearity. Then, LeCun uniform initialization is used for the weights in the first layer and Glorot uniform initialization is used for the output layers to enhance the model's ability to learn complex patterns efficiently. Besides, the output layer applies a sigmoid activation function to produce outputs that are more suitable for binary classification. Finally, the model is trained using the binary cross-entropy loss function and optimized with the Adam optimizer, with a focus on maximizing accuracy during training. The model fit with 15 epochs and a batch size of 32. The total fitting time for binary classification is about 34 minutes.



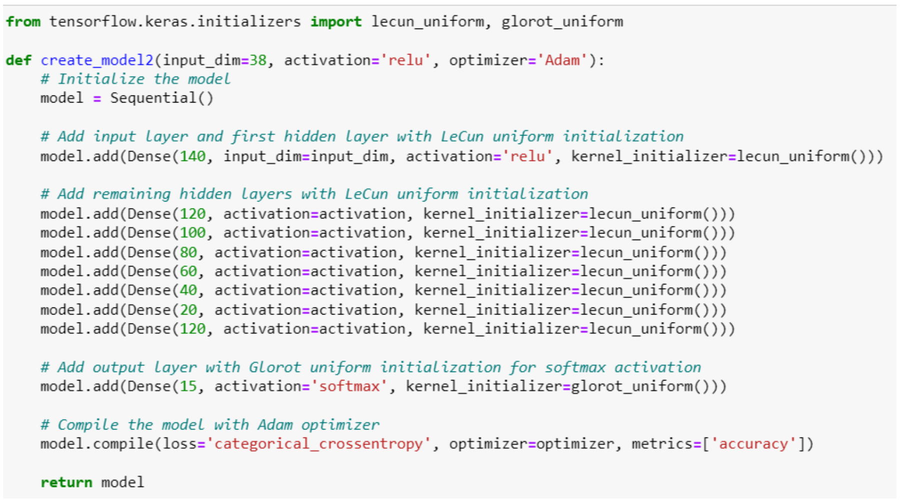
```
param_grid = {
    'batch_size': [32, 64, 128],
    'optimizer': ['adam', 'rmsprop']
}
```

```
# Perform grid search using GridSearchCV
start_time = time.time()

grid_search = GridSearchCV(estimator=model_grid, param_grid=param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)

fitting_time = time.time() - start_time
print(f"Fitting time: {fitting_time:.2f} seconds")
```

```
6807/6807 - 39s - loss: 0.0108 - accuracy: 0.9969 - 39s/epoch - 6ms/step
Epoch 7/15
6807/6807 - 42s - loss: 0.0106 - accuracy: 0.9968 - 42s/epoch - 6ms/step
Epoch 8/15
6807/6807 - 41s - loss: 0.0102 - accuracy: 0.9970 - 41s/epoch - 6ms/step
Epoch 9/15
6807/6807 - 37s - loss: 0.0096 - accuracy: 0.9973 - 37s/epoch - 5ms/step
Epoch 10/15
6807/6807 - 40s - loss: 0.0098 - accuracy: 0.9972 - 40s/epoch - 6ms/step
Epoch 11/15
6807/6807 - 31s - loss: 0.0096 - accuracy: 0.9973 - 31s/epoch - 5ms/step
Epoch 12/15
6807/6807 - 30s - loss: 0.0097 - accuracy: 0.9971 - 30s/epoch - 4ms/step
Epoch 13/15
6807/6807 - 31s - loss: 0.0091 - accuracy: 0.9974 - 31s/epoch - 4ms/step
Epoch 14/15
6807/6807 - 30s - loss: 0.0077 - accuracy: 0.9978 - 30s/epoch - 4ms/step
Epoch 15/15
6807/6807 - 31s - loss: 0.0081 - accuracy: 0.9977 - 31s/epoch - 5ms/step
Fitting time: 11965.25 seconds
```

Figure 5.5 Hyperparameter tuning on DNN model for binary classification

The same model had gone through hyperparameter tuning to find a more optimum model. The parameters tuned in this model are the batch size, which are 32, 64, and 128, and the optimizer for the input layer, which are Adam optimizer and RMSProp. The hyperparameter tuning used 3-fold cross-validation to do grid search and the total fitting time is 3.32 hours. The best parameter found for this model is a batch size of 128 with an Adam optimizer used for the input layer.



```
from tensorflow.keras.initializers import lecun_uniform, glorot_uniform

def create_model2(input_dim=38, activation='relu', optimizer='Adam'):
    # Initialize the model
    model = Sequential()

    # Add input layer and first hidden layer with LeCun uniform initialization
    model.add(Dense(140, input_dim=input_dim, activation='relu', kernel_initializer=lecun_uniform()))

    # Add remaining hidden layers with LeCun uniform initialization
    model.add(Dense(120, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(100, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(80, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(60, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(40, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(20, activation=activation, kernel_initializer=lecun_uniform()))
    model.add(Dense(120, activation=activation, kernel_initializer=lecun_uniform()))

    # Add output layer with Glorot uniform initialization for softmax activation
    model.add(Dense(15, activation='softmax', kernel_initializer=glorot_uniform()))

    # Compile the model with Adam optimizer
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

    return model
```

Figure 5.6 DNN model implementation – multiclass classification

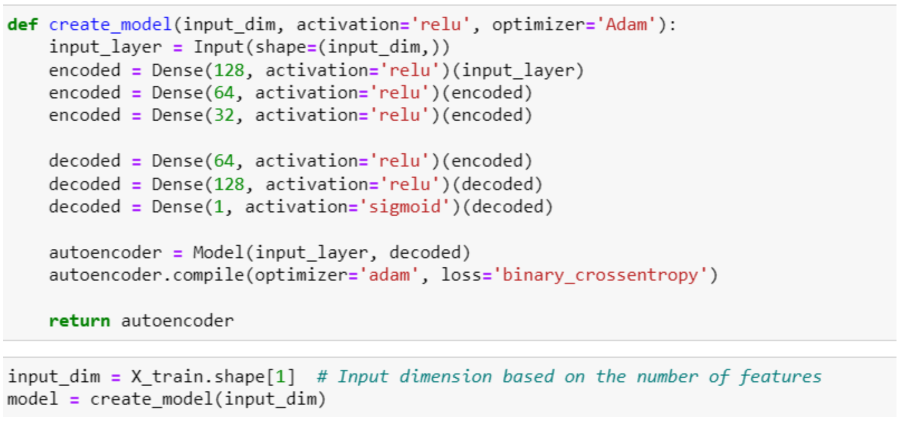
A similar DNN model had been applied for multiclass classification. However, some modifications have been done for the output layer activation and function used. The multiclass classification DNN model used Softmax as activation with 15 neurons. Then categorical cross-entropy loss is used to handle multiclass classification tasks better. The total fitting time used for this model is 34 minutes.

The similar hyperparameters used for binary classification had been applied to multiclass classification too. After using 3-fold cross-validation, the total grid search time is 3.42 hours. Different from binary class classification, the best parameters found this time is a batch size of 128 with RMSProp optimizer used for the input layer.

Table 5.2 The most optimum DNN models

Classification Type	Batch Size	Optimizer	Fitting Time (hour)
Binary Class	128	Adam	3.32
Multi Class	128	RMSProp	3.42

5.3 Deep Autoencoder Model

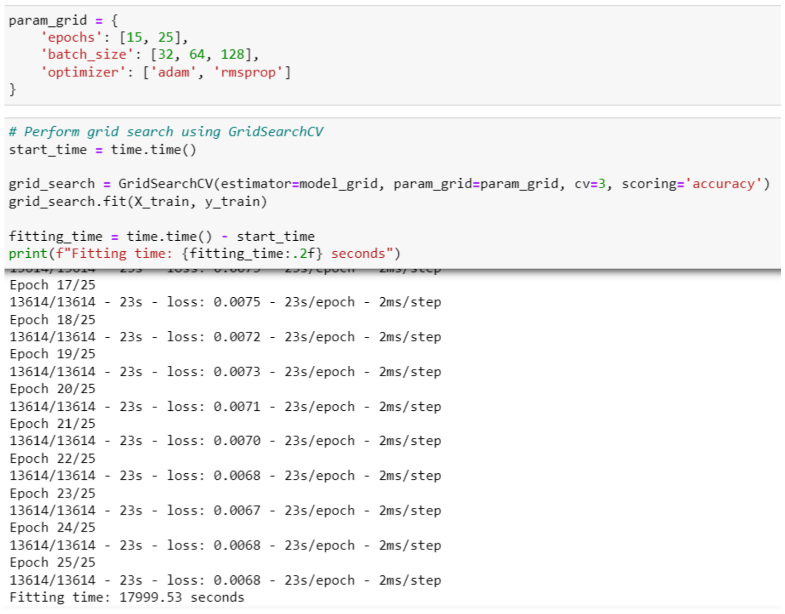


```
def create_model(input_dim, activation='relu', optimizer='Adam'):
    input_layer = Input(shape=(input_dim,))
    encoded = Dense(128, activation='relu')(input_layer)
    encoded = Dense(64, activation='relu')(encoded)
    encoded = Dense(32, activation='relu')(encoded)

    decoded = Dense(64, activation='relu')(encoded)
    decoded = Dense(128, activation='relu')(decoded)
    decoded = Dense(1, activation='sigmoid')(decoded)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

    return autoencoder
```

```
input_dim = X_train.shape[1]  # Input dimension based on the number of features
model = create_model(input_dim)
```

Figure 5.7 DAE model implementation for binary classification

The third model used is the Deep Autoencoder model. The model consists of an input layer with 38 dimensions, followed by three dense layers in the encoder part with decreasing neuron counts of 128, 64, and 32 using the ReLU activation function. Then, the decoder part mirrors the encoder’s structure, with three dense layers of 64, and 128 using the ReLU activation. The last decoder used one neuron with a sigmoid function to reconstruct the input data. The model was compiled using Adam optimizer and binary cross-entropy loss, which is more suitable for binary classification tasks. The model used 15 epochs with a batch size of 32. The fitting time for it is 27.51 minutes.



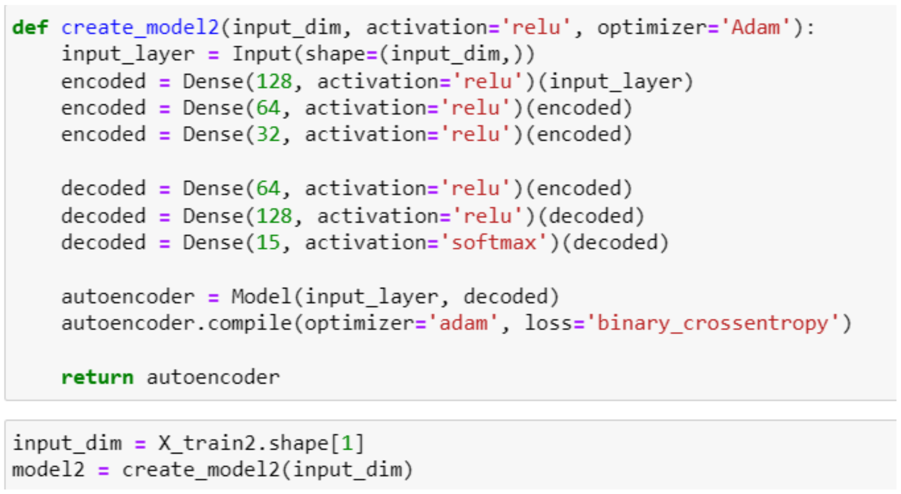
```
param_grid = {
    'epochs': [15, 25],
    'batch_size': [32, 64, 128],
    'optimizer': ['adam', 'rmsprop']
}
```

```
# Perform grid search using GridSearchCV
start_time = time.time()

grid_search = GridSearchCV(estimator=model_grid, param_grid=param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)

fitting_time = time.time() - start_time
print(f"Fitting time: {fitting_time:.2f} seconds")
```

```
Epoch 17/25
13614/13614 - 23s - loss: 0.0075 - 23s/epoch - 2ms/step
Epoch 18/25
13614/13614 - 23s - loss: 0.0072 - 23s/epoch - 2ms/step
Epoch 19/25
13614/13614 - 23s - loss: 0.0073 - 23s/epoch - 2ms/step
Epoch 20/25
13614/13614 - 23s - loss: 0.0071 - 23s/epoch - 2ms/step
Epoch 21/25
13614/13614 - 23s - loss: 0.0070 - 23s/epoch - 2ms/step
Epoch 22/25
13614/13614 - 23s - loss: 0.0068 - 23s/epoch - 2ms/step
Epoch 23/25
13614/13614 - 23s - loss: 0.0067 - 23s/epoch - 2ms/step
Epoch 24/25
13614/13614 - 23s - loss: 0.0068 - 23s/epoch - 2ms/step
Epoch 25/25
13614/13614 - 23s - loss: 0.0068 - 23s/epoch - 2ms/step
Fitting time: 17999.53 seconds
```

Figure 5.8 Hyperparameter tuning on DAE model for binary classification

The hyperparameters used for the DAE model for binary classification are epochs (15 and 25), batch size (32, 64, and 128), and optimizer (Adam and RMSProp). After using the 3-fold cross-validation, the total grid search time was estimated at five hours. The most optimum parameter found for the binary classification is a batch size of 64, epochs of 25 with RMSProp applied for encoding and decoding layers.



```
def create_model2(input_dim, activation='relu', optimizer='Adam'):
    input_layer = Input(shape=(input_dim,))
    encoded = Dense(128, activation='relu')(input_layer)
    encoded = Dense(64, activation='relu')(encoded)
    encoded = Dense(32, activation='relu')(encoded)

    decoded = Dense(64, activation='relu')(encoded)
    decoded = Dense(128, activation='relu')(decoded)
    decoded = Dense(15, activation='softmax')(decoded)

    autoencoder = Model(input_layer, decoded)
    autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

    return autoencoder
```

```
input_dim = X_train2.shape[1]
model2 = create_model2(input_dim)
```

Figure 5.9 DAE model implementation for multiclass classification

On the other hand, the DAE model applied for the multiclass classification used a Softmax optimizer with 15 neurons for the final layer, which is more suitable for multiclass tasks. The others remain the same as the model for binary class classification. After training with 15 epochs and a batch size of 32, the fitting time for the model is estimated at 30 minutes.

Similar hyperparameters are used in multiclass classification models too, which are 15 or 25 epochs, batch size of 32, 64, and 128, and optimizer of Adam and RMSProp. After the 3-fold cross-validation, the total grid search time is 7.35 hours. However, different from binary classification best parameters, the best parameter for the multiclass case is batch size with 128, epochs of 15 with RMSProp optimizer for both encode and decode layers.

Table 5.3 The most optimum DAE models

Classification Type	Epochs	Batch Size	Optimizer	Fitting Time (hour)
Binary Class	25	64	RMSProp	4.99
Multi Class	15	128	RMSProp	7.35

5.4 Summary

The study focused on implementing and optimizing three key machine learning or deep learning models for intrusion detection, which are Random Forest (RF), Deep Neural Network (DNN), and Deep Autoencoder (DAE). The initial training of RF model used ten estimators with random state of 42. Then, hyperparameter tuning had been done on both binary and multiclass classification, resulted in higher number of estimators used, 110 of maximum depth and minimum sample of 2 and 1. The fitting time for two classifications were 17.86 hours and 33.93 hours each.

The DNN model implemented input layers of 38 dimensions with a total of eight hidden layers for both binary and multi-class classification. Hyperparameter tuning had been applied which further refined the DNN models, optimizing batch sizes and optimizers to enhance performance. A batch size of 128 and optimizer of Adam and RMSProp each had been chosen for two classifications. The optimized DNN models demonstrated fitting times of 3.32 and 3.42 hours each, showcasing improved accuracy and suitability for classification tasks.

The third model, DAE model utilized autoencoder architecture with distinct encoding and decoding layers. The hyperparameter tuning on binary and multi-class classification resulted in 25 and 15 of epochs, 64 and 128 of batch size and RMSProp as optimizer for the models. The fitting times for the models are 4.99 and 7.35 hours each, indicating a notable improvement in accuracy and suitability for both classification tasks.

In summary, the study meticulously implemented, trained, and optimized RF, DNN, and DAE models for intrusion detection, for the purpose of showcasing substantial improvements in performance metrics such as accuracy and fitting times through rigorous hyperparameter tuning and model refinement processes.

Chapter 6

Experimental Result and Discussion

6.1 Pre-processing Results

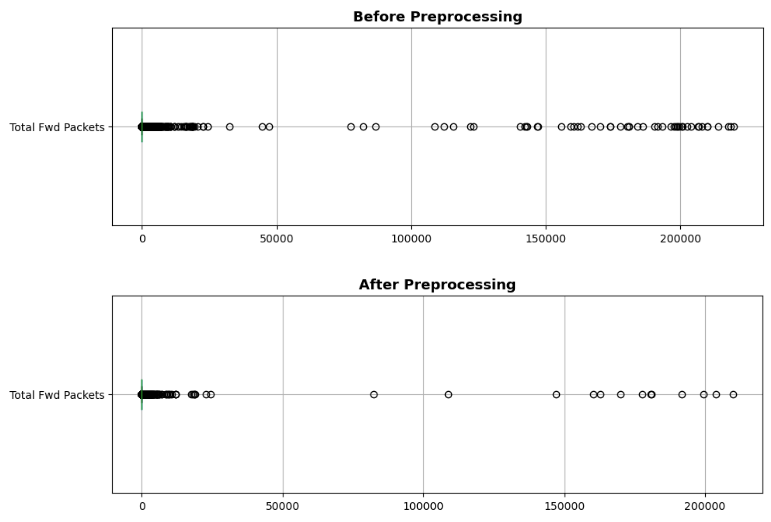


Figure 6.1.1 Box Plot comparison of one feature

The comparison of the box plots for one of the features in the CIC-IDS2017 dataset before and after pre-processing reveals a significant reduction in outliers, particularly in the total forward packets. Initially, the data showed numerous outliers ranging from 150,000 to 200,000, but after pre-processing, only a few outliers remained. While the pre-processing didn't eliminate all outliers, it was highly effective in reducing their presence in the dataset, leading to a more refined and reliable data distribution.

6.2 Experimental Results – Random Forest

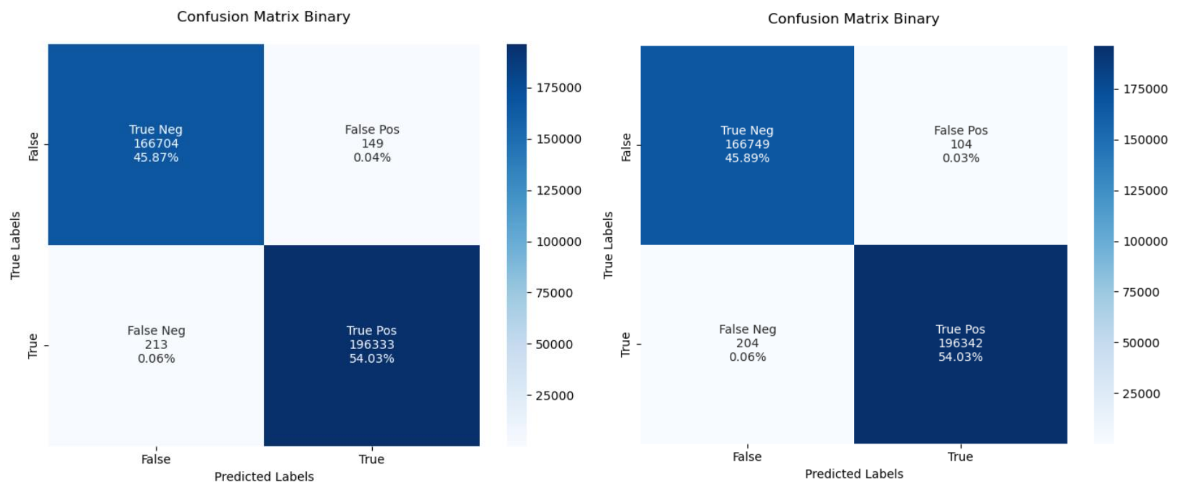


Figure 6.2.1 RF model confusion matrix for binary classification (Before and After Hyperparameter Tuning)

The above comparison shows the confusion matrix results for Random Forest (RF) models before and after hyperparameter adjustment in binary classification. The benefit of hyperparameter adjustment is noticeable in the reduction of false positives, which dropped from 0.04% to 0.03%. This drop indicates a significant improvement in the models' capacity to reduce false alarms, increasing the precision of their forecasts.

Similarly, the tuning process resulted in a decrease in false negatives, from 213 to 204. This decrease in false negatives suggests that the model's sensitivity and recall have improved, resulting in an overall gain in accuracy. By efficiently resolving both false positives and false negatives, hyperparameter-tuned RF models increase their performance in correctly identifying instances of both normal and attack traffic in the dataset.

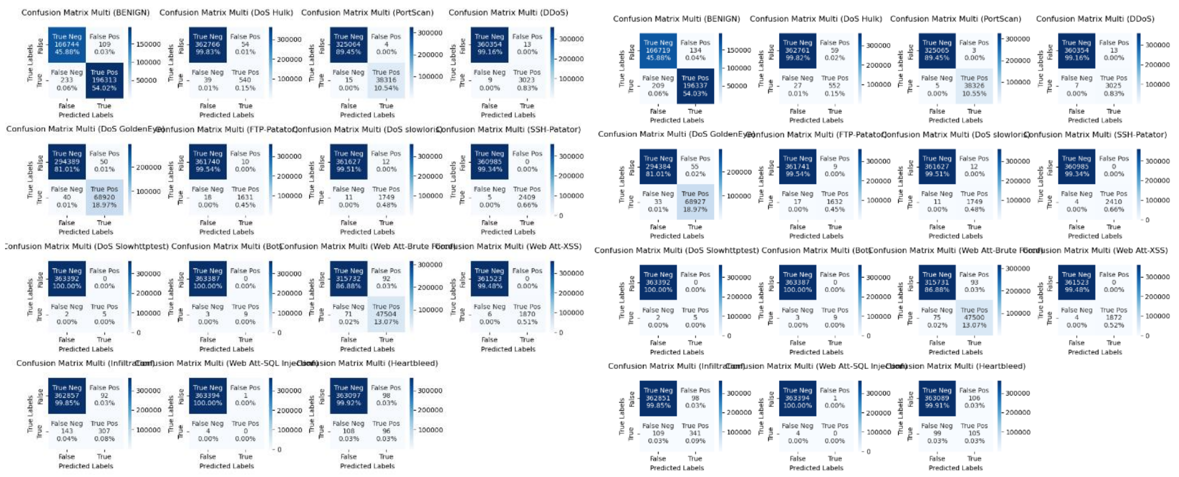


Figure 6.2.2 RF model confusion matrix for multi-class classification (Before and After Hyperparameter Tuning)

The comparison of the confusion matrices before and after hyperparameter adjustment for multi-class classification revealed little differences. Overall, True Positive rates for most attack categories increased slightly, ranging from at least one sample to 0.1% of total samples. These categories include DoS, Port Scan, DDoS, FTP, and SSH, among others. This development suggests that the models have become more adept at correctly identifying these specific types of attacks.

However, an increase in False Positive rates for the RF model was found, which rose from one sample to 0.1% of the total data. This rise indicates a risk of false alerts for specific attack types. Although there was an increment in True Positive rates, the increase in False Positives implies a need for further refining to reduce false alarm rates and improve the precision of the multi-class classification models.

6.3 Experimental Results – Deep Neural Network

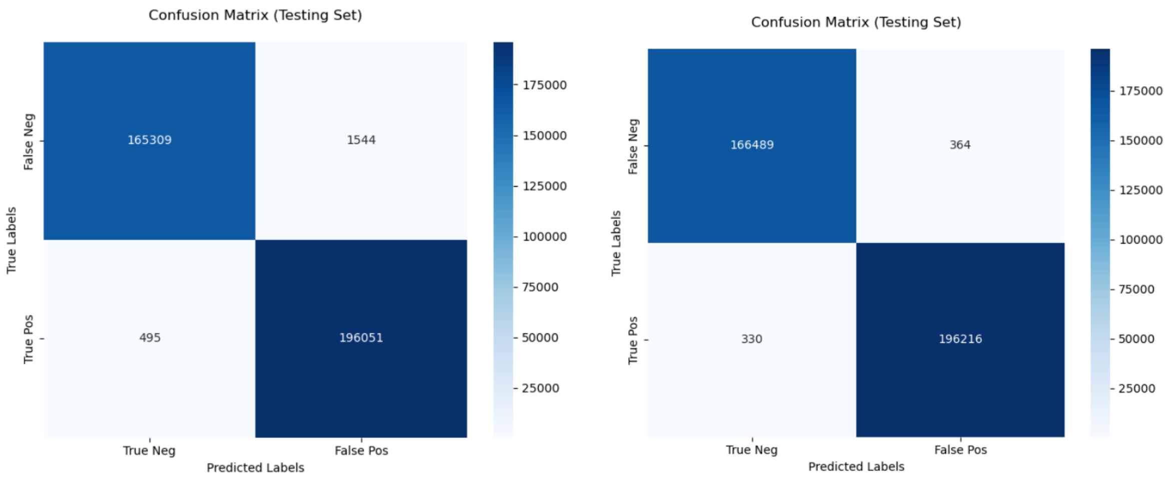


Figure 6.3.1 DNN model confusion matrix for binary classification (Before and After Hyperparameter Tuning)

The DNN model performed better after hyperparameter adjustment. Notably, false positives decreased significantly, from 1544 to 364 samples. This huge reduction implies that the model's false alarm rate has decreased significantly, increasing its reliability in accurately identifying normal traffic.

Furthermore, the DNN model's false negatives fell by 165 samples, while true negatives and true positives increased by 1180 and 165 samples, respectively. After hyperparameter optimization, the DNN model's prediction power becomes more balanced and accurate, as seen by these changes.

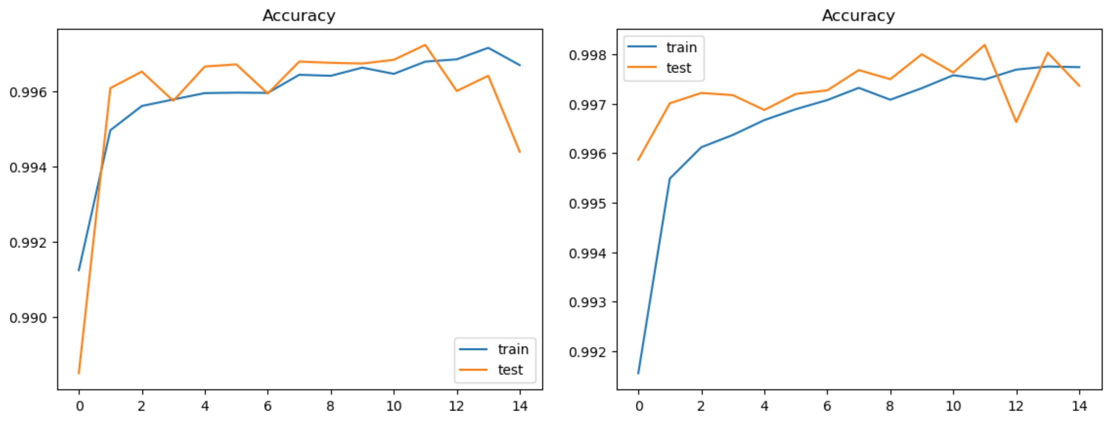


Figure 6.3.2 DNN model accuracy curve for binary classification (Before and After)

The graphs displaying the accuracy of the Deep Neural Network (DNN) model for both the training and testing sets show a clear trend of improvement as the epoch count increases. The greatest accuracy value had shifted from 0.996 to 0.998 before and after hyperparameter adjustment. Initially, the accuracy for the testing set is around 0.996 and gradually increases to around 0.998. This incremental increase suggests that as the model progresses through more epochs, it improves its ability to learn from training data and generalize its predictions to previously unseen data in the testing set.

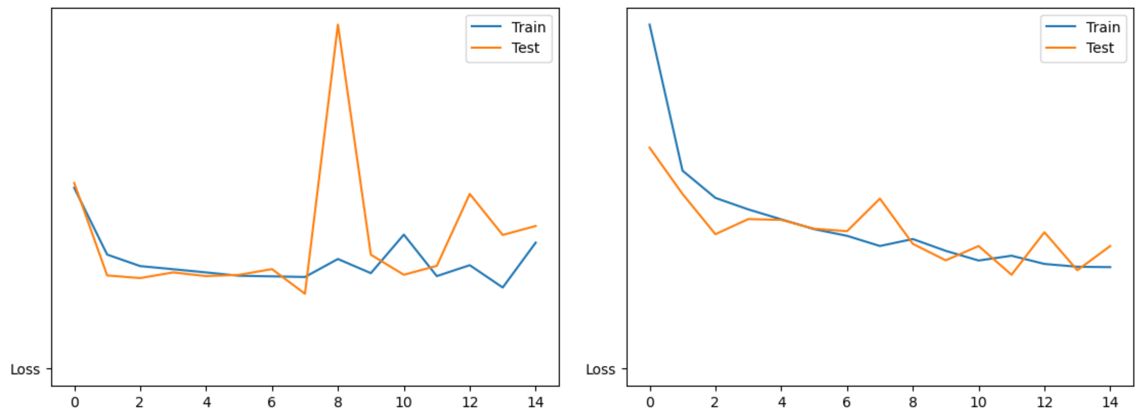


Figure 6.3.3 DNN model loss curve for binary classification (Before and After)

The Deep Neural Network (DNN) model's loss curve provides critical insights into its performance throughout training and testing, especially after hyperparameter tuning. Initially, the loss curve may show oscillates or unexpected spikes, suggesting places where the model struggles to converge or has difficulty reducing the loss function.

However, after hyperparameter adjustment, the loss curve shows significant improvement. One noticeable improvement is the graph's overall smoothness, which indicates that the model's training and testing processes have become more stable and reliable. This smoother curve indicates that the model is now better able to learn and generalize from the data, resulting in more trustworthy predictions and less fluctuation in performance.

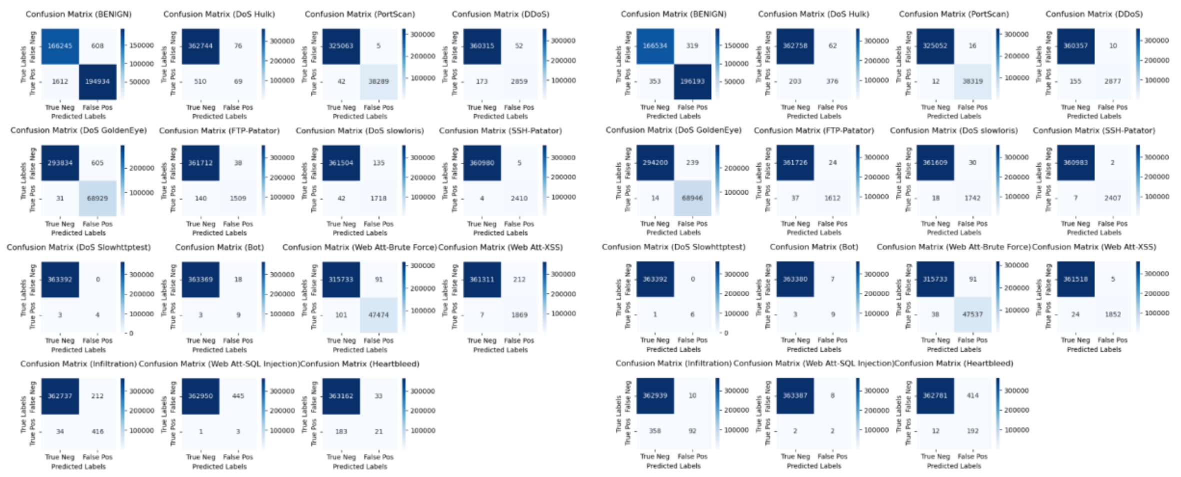


Figure 6.3.4 DNN model confusion matrix for multi-class classification (Before and After Hyperparameter Tuning)

The confusion matrix study of the DNN model for multi-class classification shows considerable gains in identifying various types of attacks. Several attack types, including as DoS, Port Scan, DDoS, FTP, SSH, Brute Force, and Heartbleed, saw significant improvements in their confusion matrices. These gains were distinguished by a decrease in false positives and negatives, as well as an increase in real positives and negatives. These changes combined show a significant improvement in the model's accuracy and reliability in detecting these attack types.

However, several categories, such as XSS, Infiltration, and SQL Injection, witnessed a performance decrease. This reduction is demonstrated by a decrease in the frequency of true positives, implying that the model's ability to reliably categorize these specific attacks may have declined following hyperparameter adjustment.

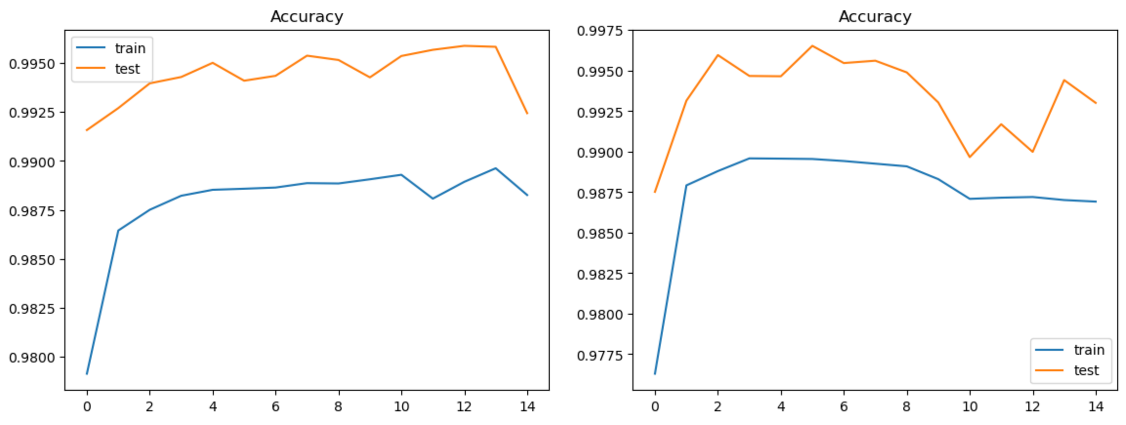


Figure 6.3.5 DNN model accuracy curve for multi-class classification (Before and After)

The accuracy curve analysis for the DNN model after hyperparameter tuning shows an improvement of around 0.995 to 0.997. This upgrade implies that the model's accuracy has improved generally after fine-tuning. However, an interesting observation is made in the curve for the testing set, which appears less smooth than in its pre-tuning form. In contrast, the training set's curve remains rather stable both before and after tuning.

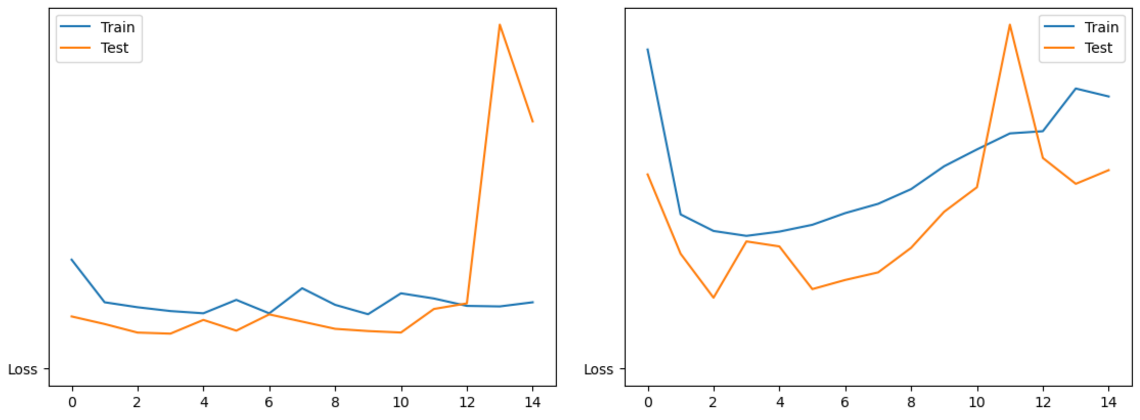


Figure 6.3.6 DNN model loss curve for multi-class classification (Before and After)

The loss curve study for the DNN model in multi-class classification shows a less smooth trend than binary classification. Additionally, there is a noticeable increase in loss following hyperparameter adjustment. This issue could be related to the model's extended goal of predicting numerous classes, which may make it difficult to efficiently minimize the loss function.

The increase in loss following hyperparameter tuning indicates that the model is having difficulty reliably classifying over a wider range of categories. This could be due to greater complexities and variability in the multi-class categorization scenario, forcing the model to negotiate a more difficult decision environment.

6.4 Experimental Results – Deep Autoencoder

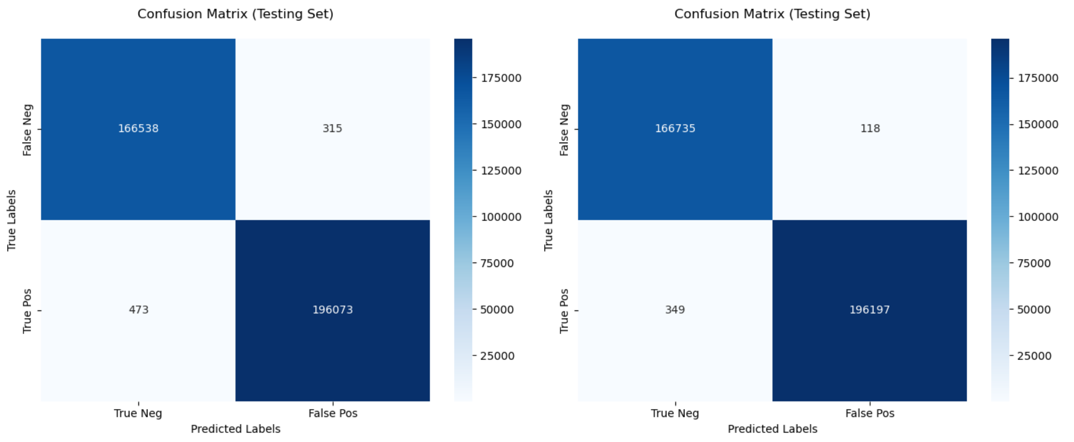


Figure 6.4.1 DAE model confusion matrix for binary classification (Before and After Hyperparameter Tuning)

The binary classification analysis for the DAE model reflects the improvements seen in the RF and DNN models following hyperparameter adjustment. Notably, performance metrics have improved significantly, with false positive and false negative cases down by 197 and 124, respectively. Furthermore, true positive cases climbed from 196073 to 196197, while true negative cases rose from 166538 to 166735.

These enhancements result in increased accuracy and precision for the DAE model in binary classification tasks. The decrease in false positives and false negatives represents a drop in the model's false alarm rate and misclassification errors, contributing to improved overall performance and reliability in discriminating between benign and attacked traffic.

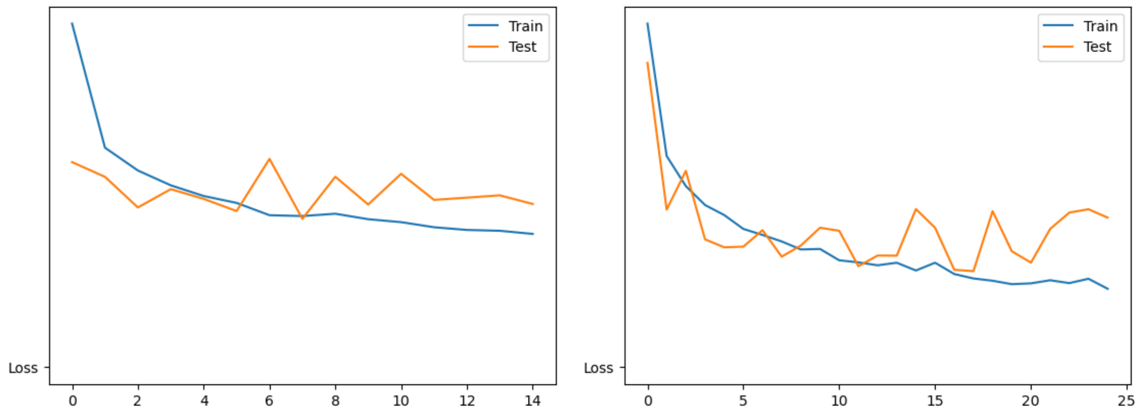


Figure 6.4.2 DAE model loss curve for binary classification (Before and After)

The loss curve analysis before and after hyperparameter tuning shows a significant improvement, as seen by a reduction in loss for both training and testing sets. This drop in loss indicates that the model's prediction accuracy and capacity to minimize errors have improved after tuning. Furthermore, the training set shows a smoother curve after tuning, especially with additional epochs used during fitting. This smoother curve suggests increased model performance as more training iterations occur, implying better convergence and optimization of the model’s learning process..

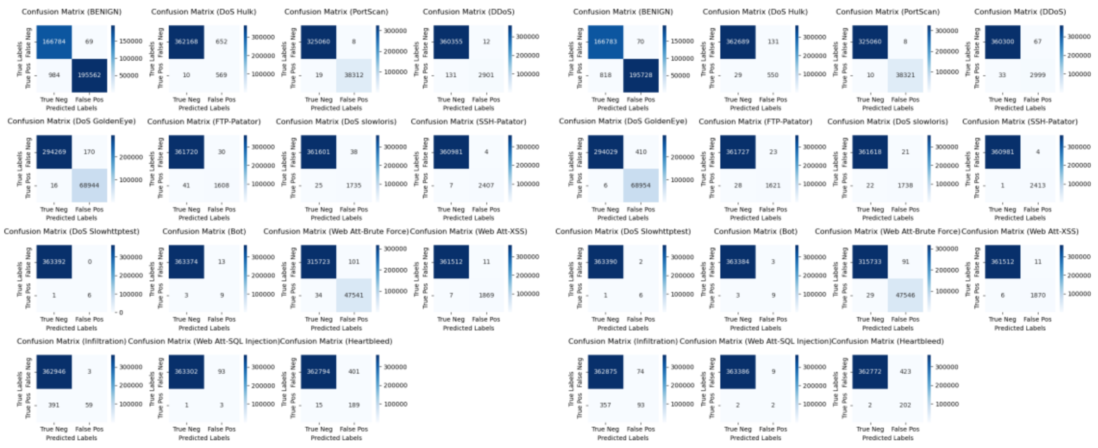


Figure 6.4.3 DAE model confusion matrix for multi-class classification (Before and After Hyperparameter Tuning)

The multi-class classification analysis of the DAE model shows a moderate improvement after hyperparameter adjustment. While some categories failed to effectively minimize false positive results, resulting in a larger false alarm rate, other areas showed significant improvement. Several categories saw a decrease in false negatives and an increase in true positives inside the confusion matrix. These modifications suggest an overall improvement in the model's accuracy and performance, especially in correctly identifying occurrences of different attack types.

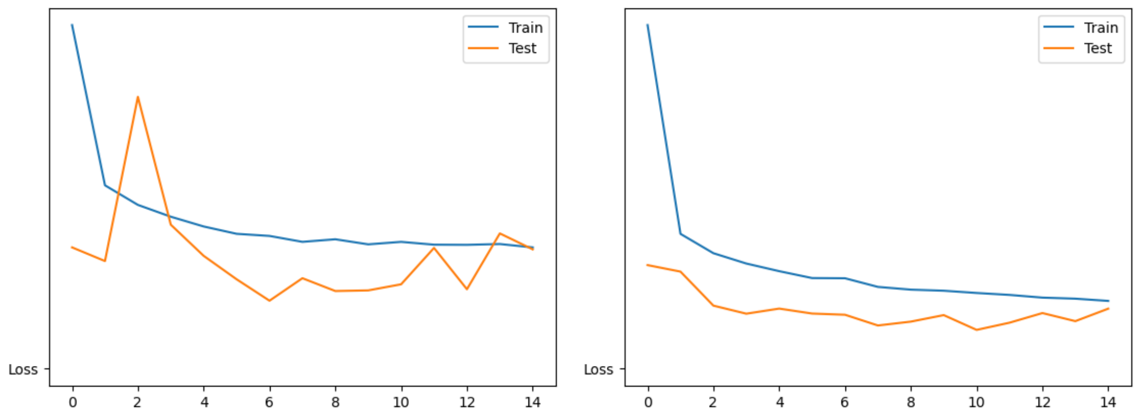


Figure 6.4.4 DAE model loss curve for multi-class classification (Before and After)

The loss curve for the DAE model in multi-class classification improves significantly after hyperparameter optimization, resulting in a smoother and more stable graph for the test set. This smoother trend shows improved model performance and lower variability in loss values. Furthermore, the general trend of the loss curve displays a significant reduction in loss relative to the curve before hyperparameter adjustment, indicating the model's increased effectiveness in minimizing mistakes during the training and validation stages.

6.5 Comparison of Performance with Previous Researchers

Below is the comparison of the results of each machine learning and deep learning before hyperparameter tuning, after hyperparameter tuning, and the other actors' results:

Table 6.5.1 Results of models for Binary Classification

Model	Dataset	Binary Classification (%)			
		Accuracy	Precision	Recall	F1 Score
RF	CIC-IDS2017	99.90	99.90	99.90	99.90
RF (Tuned)	CIC-IDS2017	99.92	99.92	99.92	99.92
RF-800 (Rifkie P. & Bayu A. T.) [2017]	NSL-KDD	99.57	-	-	-
RF (Adnan H. A., Salama A. M. & etc.) [2021]	CIC-IDS2017	96.76	97.96	97.62	-

DNN + 8 hidden layers	CIC-IDS2017	99.44	99.44	99.44	99.44
DNN + 8 hidden layers (Tuned)	CIC-IDS2017	99.81	99.81	99.81	99.81
Feature Selection Based DNN (Li H. L., Ramli A., Wen C. T., & Alok K. S.) [2021]	NSL-KDD	99.40	99.70	97.90	98.80
DAE	CIC-IDS2017	99.78	99.78	99.78	99.78
DAE (Tuned)	CIC-IDS2017	99.87	99.87	99.87	99.87
DAE (Srikanthyadav M. Gayatri K. Padmaja R.)	CIC-IDS2017	91.76	96.27	96.40	-

In contrast to the Random Forest model utilized by earlier researchers, the Random Forest model in this study report performed better in binary classification, with accuracy, precision, recall, and an F1 score of 99.90. There was a 0.02% boost in performance after tuning the model. The performance is because research conducted feature selection and dataset pre-processing. About 0.35% of the performance was successfully boosted by the research.

Conversely, the study employed a DNN model with eight hidden layers, similar to Petros Toupas's model. The model has a 99.44% accuracy, precision, recall, and F1 score, a percentage significantly higher than the binary classification result provided by Li Hua and others. Performance improved by 0.35% when the hyperparameters were adjusted, with an accuracy difference of up to 0.41%.

Finally, the DAE model trained by this research succeeded at 99.78%. After hyperparameter adjustment, performance improved to 99.87%. Compared to the performance results provided by previous researchers, this study boosted the accuracy of the DAE model by 8.11%, the highest among the RF and DNN models.

In conclusion, the study paper effectively trained the model to obtain at least 99% accuracy, precision, recall, and F1 score. There was 99% confidence in IDS trained using Random Forest, DNN, and DAE models to discern between benign and attack traffic, as provided by the hyperparameters employed in this study paper and feature selection done on the dataset fed in.

Table 6.5.2 Results of models for Multi-class Classification

Model	Dataset	Multi-class Classification (Average) (%)			
		Accuracy	Precision	Recall	F1 Score
RF	CIC-IDS2017	99.81	99.85	99.81	99.83
RF (Tuned)	CIC-IDS2017	99.83	99.84	99.83	99.84
RF (Nabila F. & M. A. Jabbar) [2016]	NSL-KDD	99.67	-	-	-
DNN + 8 hidden layers	CIC-IDS2017	99.21	99.36	99.21	99.26
DNN + 8 hidden layers (Tuned)	CIC-IDS2017	99.66	99.72	99.66	99.65
DNN + 8 hidden layers (Petros T., Dimitra C. & etc.) [2019]	CIC-IDS2017	99.95	94.31	95.62	94.10
DAE	CIC-IDS2017	99.54	99.75	99.54	99.58
DAE (Tuned)	CIC-IDS2017	99.63	99.68	99.63	99.63
DAE (Yesi N., Kunang, Siti N., Deris S., & Bhakti Y. S.)	CIC-IDS2018	95.79	95.38	95.79	95.11

For this study's multi-class classification, the Random Forest model achieved 99.81% accuracy, 99.85% precision, 99.81% recall, and a 99.83% F1 score. After hyperparameter adjustment, the model's performance improved somewhat, with an accuracy of 99.83%, precision of 99.84%, recall of 99.83%, and F1 score of 99.84%. However, using feature selection, the model was able to boost its accuracy by 0.16%.

Aside from that, the DNN model achieved an accuracy of 99.21%, precision of 99.36%, recall of 99.21%, and F1 score of 99.26%. After hyperparameter adjustment, the accuracy, precision, recall, and F1 score increased to 99.65%. Even though the DNN model provided by our research had a lower accuracy than the model proposed by Petros et al., the precision, recall, and F1 scores were substantially greater, around 5% higher. Thus, the DNN model developed in this study exhibited a more balanced performance, with a high detection rate and a low false alarm rate.

Finally, the DAE model developed in this study achieved an accuracy of 99.54%, a precision of 99.75%, a recall of 99.54%, and an F1 score of 99.58%. After hyperparameter adjustment, the DAE model's accuracy rose by roughly 0.9%. In contrast to the DAE model offered by Yesi, etc., the research successfully boosted performance by approximately 4%. The increment could be because the DAE model employed in the study differs from that used by prior researchers, as earlier researchers did not produce many results for the deep learning model. In addition, an appropriate pre-processing with feature selection was applied to the dataset before training.

In summary, the models trained in this study, which include an RF model, a DNN model, and a DAE model, were able to identify 15 categories of activity types in the CIC-IDS2017, including benign and 14 forms of attacks such as DoS, DDoS, Port Scan, and others. The attack rate exceeds 99%, with the DAE model having the lowest accuracy (99.63%) and the RF model having the best accuracy (99.83%). On the other hand, the DAE model has the lowest recall of 99.63%, followed by DNN at 99.66% and RF at 99.83%. There is more than 99.6% confidence that the models, following hyperparameter adjustment, will not produce a false alarm rate or precise detection.

6.6 Project Challenges

Nonetheless, it's critical to recognize the study approach's limitations. Time constraints prevent the research from simulating our data and updating the models to cater to evolving attack tactics. Furthermore, considering that using more hyperparameters during Grid Search will lengthen the time required to identify the ideal model and improve performance, the research only selected the bare minimum of hyperparameters, guaranteeing that the findings will be trained in two days.

Overall, the study demonstrated the potential optimization strategies and hyperparameters used in improving the accuracy, precision, recall, and F1 scores of the RF, DNN, and DAE models.

6.7 Summary

This chapter presented experimental data for the RF model, DNN model, and DAE model. In binary classification tasks, hyperparameter adjustment improved model performance across all three models, with large decreases in false positives and false negatives. The DNN model, in particular, demonstrated a reduction in false positives and a balanced improvement in real positives and negatives, resulting in increased precision and recall.

The RF model performed well in multi-class classification, with just minimal post-tuning changes needed. After hyperparameter adjustment, the DNN model performed more consistently, with improved accuracy and precision. The DAE model also demonstrated moderate gains, particularly in terms of reducing false negatives and boosting true positives, showing improved detection accuracy across attack types.

The feature selection approach used on the CIC-IDS2017 dataset, together with model hyperparameter adjustments, significantly enhanced the performance of the Random Forest, Deep Neural Network, and Deep Autoencoder models. These enhancements resulted in outstanding accuracy rates of up to 99.5% for both binary and multi-class classifications, implying that the IDS system can detect around 99.5% of attacked traffic. Furthermore, reaching high precision and recall rates of up to 99.5% illustrates the models' capacity to maintain a low false alarm rate while reliably detecting malicious activity. These results demonstrate that these models can be used effectively in intrusion detection system (IDS) environments, given that the appropriate pre-processing, feature selection, and hyperparameter tuning are done.

Furthermore, the obstacles encountered during the study, such as time constraints that limited the exploration of varied datasets and hyperparameters, were acknowledged. Despite these restrictions, the study demonstrated successful optimization methodologies and hyperparameter tuning that significantly enhanced model performance across a range of expectations. This thorough approach not only highlights the models' efficacy but also stresses the significance of meticulous data preprocessing and model refinement in generating robust results in IDS applications.

Chapter 7
Conclusion

7.1 Conclusion

The research had done a comprehensive approach toward improving machine learning models for intrusion detection systems. The study focused on critical processes such as dataset pre-processing, feature selection, model analysis, and hyperparameter tuning, with each step having the goal of improving model performance across binary and multiclass classification tasks using the CIC-IDS2017 dataset. The first phases involved fine-tuning the dataset, addressing difficulties such as null values, data merging, attribute reduction, label standardization, data balance, scaling, encoding, and partitioning for training and testing. Feature selection, guided by Pearson's Correlation, focused on critical properties, creating a solid foundation for later model fitting. The research focused on assessing and fine-tuning machine learning models such as Random Forest (RF), Deep Neural Network (DNN), and Deep Autoencoder (DAE). Performance parameters such as accuracy, precision, recall, and F1 score were analysed to determine each model's ability to reliably detect normal and attack traffic.

Hyperparameter tuning emerged as a critical step, resulting in significant performance improvements. After tuning, the models had outstanding accuracies that were regularly higher than 99.5%, with the RF model leading the group by achieving a 99.92% average in binary classification. Notably, the DNN model in multi-class classification achieved an optimal balance of precision (99.72%) and recall (99.66%), demonstrating its capacity to detect attack traffic while minimizing false alarms. This study provides significant insights and strategies for designing robust IDS solutions that have improved detection capabilities and lower false alarm rates, strengthening network security in modern cyber environments.

7.1 Recommendation

Despite the project's success, time constraints limited hyperparameter changes to a subset of features. Aside from that, the research was unable to develop our dataset to use in this study. Future efforts should focus on broader hyperparameter tuning approaches to investigate the full range of model optimization options. This includes altering hyperparameters such as learning rates, regularization parameters, and network designs across all important aspects to ensure a thorough examination of the model's capabilities. Furthermore, the research should look at advanced feature engineering approaches other than Pearson's Correlation to improve model interpretability and performance by applying them to our dataset for more complete results. Overall, this study provides the foundation for reliable intrusion detection systems by proving the efficacy of precise data handling, strategic feature selection, and tailored model optimization in developing high-accuracy security solutions.

REFERENCES

[1] Rapid7, “The Pros & Cons of Intrusion Detection Systems,” Rapid7. [Online]. Available: https://www.rapid7.com/blog/post/2017/01/11/the-pros-cons-of-intrusion-detection-systems/

[2] S. Singh, “Importance of Pre-Processing in Machine Learning,” KDnuggets. [Online]. Available: https://www.kdnuggets.com/2023/02/importance-preprocessing-machine-learning.html

[3] A. Bajaj, “Performance Metrics in Machine Learning [Complete Guide],” neptune.ai. [Online]. Available: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide#:~:text=Metrics%20are%20used%20to%20monitor

[4] S. Pandian, “A Comprehensive Guide on Hyperparameter Tuning and its Techniques,” Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/

[5] UNB, “IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” UNB. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[6] R. Primartha and B. A. Tama, “Anomaly detection using random forest: A performance revisited,” in 2017 International Conference on Data and Software Engineering (ICoDSE), pp. 1–6. doi: https://doi.org/10.1109/ICODSE.2017.8285847.

[7] A. H. Azizan et al., “A Machine Learning Approach for Improving the Performance of Network Intrusion Detection Systems,” Annals of Emerging Technologies in Computing, vol. 5, no. 5, pp. 201–208, Mar. 2021, doi: https://doi.org/10.33166/aetic.2021.05.025.

[8] N. Farnaaz and J. Akhil, “Random Forest Modeling for Network Intrusion Detection System,” Procedia Computer Science, vol. 89, pp. 213–217, Dec. 2016, doi: https://doi.org/10.1016/j.procs.2016.06.047.

[9] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, “An Intrusion Detection System for Multiclass Classification Based on Deep Neural Networks,” in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 1253–1258. doi: https://doi.org/10.1109/ICMLA.2019.00206.

[10] L. H. Li, R. Ahmad, W. C. Tsai, and A. K. Sharma, “A Feature Selection Based DNN for Intrusion Detection System,” in 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), pp. 1–8. doi: https://doi.org/10.1109/IMCOM51814.2021.9377405.

[11] S. Moraboena, G. Ketepalli, and P. Ragam, “A Deep Learning Approach to Network Intrusion Detection Using Deep Autoencoder,” Revue d’Intelligence Artificielle, vol. 34, no. 4, pp. 457–463, Sep. 2020, doi: https://doi.org/10.18280/ria.340410.

[12] Kunang, Yesi Novaria, S. Nurmaini, D. Stiawan, and Suprapto, Bhakti Yudho, “Attack classification of an intrusion detection system using deep learning and hyperparameter

optimization,” Journal of Information Security and Applications, vol. 58, p. 102804, 2021, doi: https://doi.org/10.1016/j.jisa.2021.102804.

[13] E. Jaw and X. Wang, “Feature Selection and EnsembleBased Intrusion Detection System: An Efficient and Comprehensive Approach,” Symmetry, vol. 13, no. 10, 2021, doi: https://doi.org/10.3390/sym13101764.

[14] I. Karna, A. Madam, C. Deokule, R. Adhao, and V. Pachghare, “EnsembleBased Filter Feature Selection Technique for Building FlowBased IDS,” in 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), pp. 324–328. doi: https://doi.org/10.1109/ACCESS51619.2021.9563297.

[15] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” Neurocomputing, vol. 415, pp. 295–316, 2020, doi: https://doi.org/10.1016/j.neucom.2020.07.061.

[16] Analytics Vidhya, “9 Feature Transformation & Scaling Techniques| Boost Model Performance,” Analytics Vidhya, [Online]. Available: https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/

[17] IBM, “What is Random Forest? | IBM,” IBM, [Online]. Available: https://www.ibm.com/topics/random-forest

[18] C. Joo et al., “Machinelearningbased optimization of operating conditions of naphtha cracking furnace to maximize plant profit,” in 33 European Symposium on Computer Aided Process Engineering, Elsevier, 2023, pp. 1397–1402. doi: https://doi.org/10.1016/B9780443152740.502225.

[19] Deci, “What is Deep Neural Network (DNN)?,” Deci, [Online]. Available: https://deci.ai/deep-learning-glossary/deep-neural-network-dnn/

[20] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” Neurocomputing, vol. 234, pp. 11–26, 2017, doi: https://doi.org/10.1016/j.neucom.2016.12.038.

[21] M.-J. Kang and J.-W. Kang, “Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security,” PLOS ONE, vol. 11, no. 6, p. e0155781, Jun. 2016, doi: https://doi.org/10.1371/journal.pone.0155781.

[22] A. Bhandari, “Confusion matrix for machine learning,” Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

[23] S. K. Agrawal, “Evaluation Metrics For Classification Model | Classification Model Metrics,” Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

[24] Jupyter, “The Jupyter Notebook — Jupyter Notebook 6.1.1 documentation,” Jupyter [Online]. Available: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 1&2**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE [Please write the details of the work done in the last fortnight.] - Reorganized the work to be done and parts to be improved in the report.
2. WORK TO BE DONE - Re-train the model with hyperparameter tuning. - Re-write contribution and literature review
3. PROBLEMS ENCOUNTERED - Find out the models and tuning method to be done.
4. SELF EVALUATION OF THE PROGRESS - I will try to re-modify my contribution and literature review as discussed.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 3,4&5**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Trained on Random Forest Model and GNB model with hyperparameter tuning

2. WORK TO BE DONE

- Enhance the training method using Grid Search or Random Search
- Find out other models to be trained

3. PROBLEMS ENCOUNTERED

- Models' performance not very good.

4. SELF EVALUATION OF THE PROGRESS

- The training process is not smooth and performance not as good. I will try to study through the deep learning to find out more possible model to be trained.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 6,7&8**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Tried out model training on RF model and DNN model

2. WORK TO BE DONE

- Re-train the DNN model
- Find out previous researchers' work to be compared
- Write Chapter 1

3. PROBLEMS ENCOUNTERED

- Unsure about the output to be included into report
- DNN model did not perform better in both before and after hyperparameter tuning.
- Not fully understand about deep learning

4. SELF EVALUATION OF THE PROGRESS

- I am trying to many possibilities of models which caused me to lose track from my research. I need to re-understand on my contribution and train the models according to my objectives.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	**Study week no.: 9&10**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Completed modifying the DNN model with hyperparameter tuning
- Completed Chapter 1

2. WORK TO BE DONE

- Train one more model
- Write Chapter 2 and 3

3. PROBLEMS ENCOUNTERED

- Find out the possible model that can be trained.
- Previous Researchers' Work to be compared with the new model.

4. SELF EVALUATION OF THE PROGRESS

- I will try to run the new test on new model.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	**Study week no.: 11**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE [Please write the details of the work done in the last fortnight.] - Completed the third model training and hyperparameter tuning. - Completed Chapter 2 and 3
2. WORK TO BE DONE - Write Chapter 4, 5 and 6
3. PROBLEMS ENCOUNTERED - Add box plot to pre-processing steps.
4. SELF EVALUATION OF THE PROGRESS - I will try to finish the chapters above in next week.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

- Completed Chapter 4, 5, and 6

2. WORK TO BE DONE

- Write Chapter 7 and format the report
- Modify Chapter 2, 3, 4, 5, and 6
- Do poster
- Do presentation slides

3. PROBLEMS ENCOUNTERED

- Literature review not comprehensive enough and need to add more.
- Need to add summary for each chapter.

4. SELF EVALUATION OF THE PROGRESS

- During modifying of report, found out that the Chapter 6 written still able to be improved. It shall be done by next week together with modification of other chapters.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	**Study week no.: 13**
Student Name & ID: Tan May May, 20ACB01863	
Supervisor: Cik Zanariah binti Zainudin	
Project Title: INTRUSION DETECTION SYSTEM (IDS) USING MACHINE LEARNING	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

- Completed report
- Completed poster

2. WORK TO BE DONE

- Do presentation slides

3. PROBLEMS ENCOUNTERED

- No problem encountered

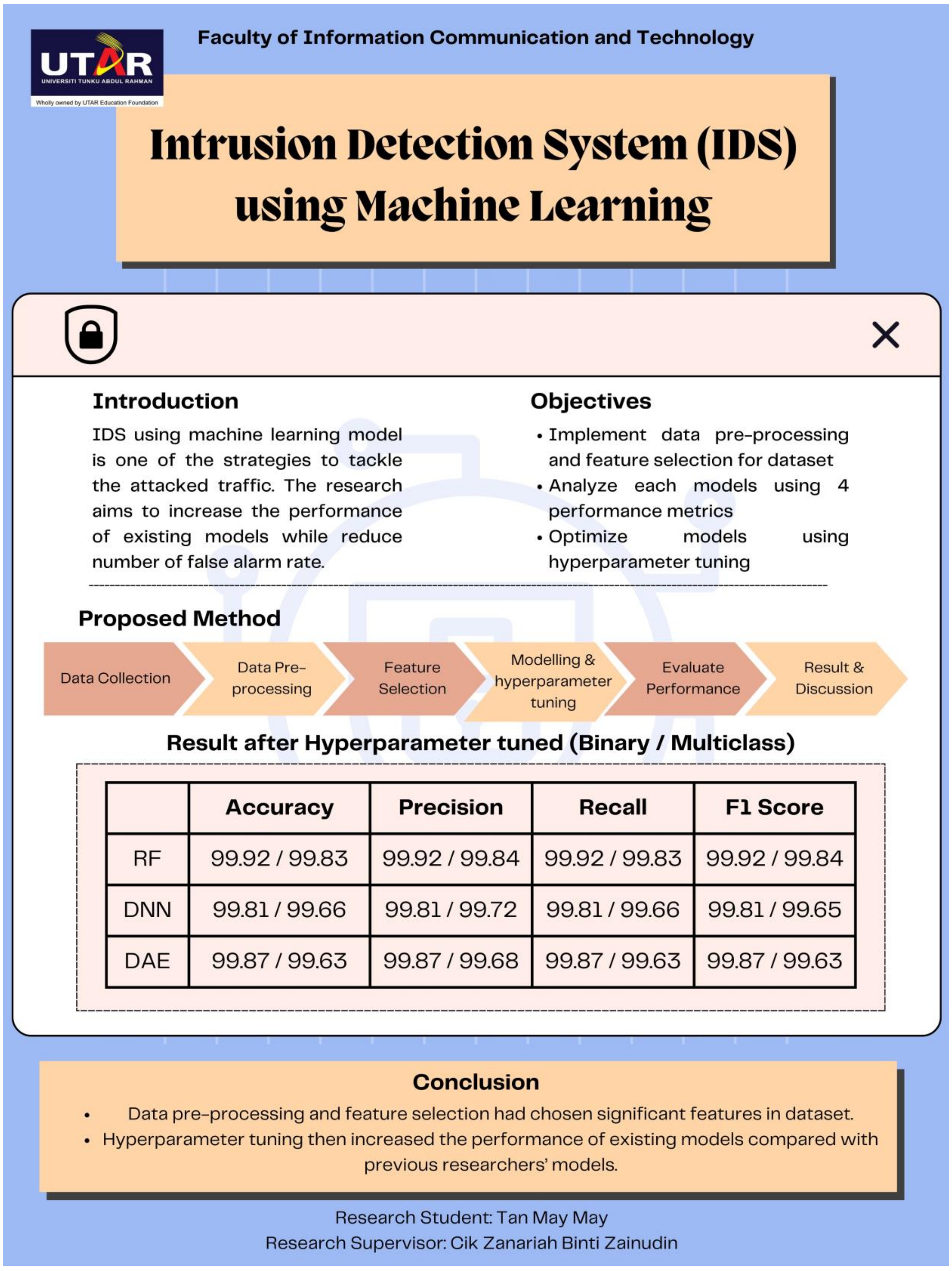
4. SELF EVALUATION OF THE PROGRESS

- I shall prepare well for the presentation.

[illegible]
Supervisor's signature

[illegible]
Student's signature

POSTER



Faculty of Information Communication and Technology

Intrusion Detection System (IDS) using Machine Learning

Introduction

IDS using machine learning model is one of the strategies to tackle the attacked traffic. The research aims to increase the performance of existing models while reduce number of false alarm rate.

Objectives

- Implement data pre-processing and feature selection for dataset
- Analyze each models using 4 performance metrics
- Optimize models using hyperparameter tuning

Proposed Method

Data Collection → Data Pre-processing → Feature Selection → Modelling & hyperparameter tuning → Evaluate Performance → Result & Discussion

Result after Hyperparameter tuned (Binary / Multiclass)

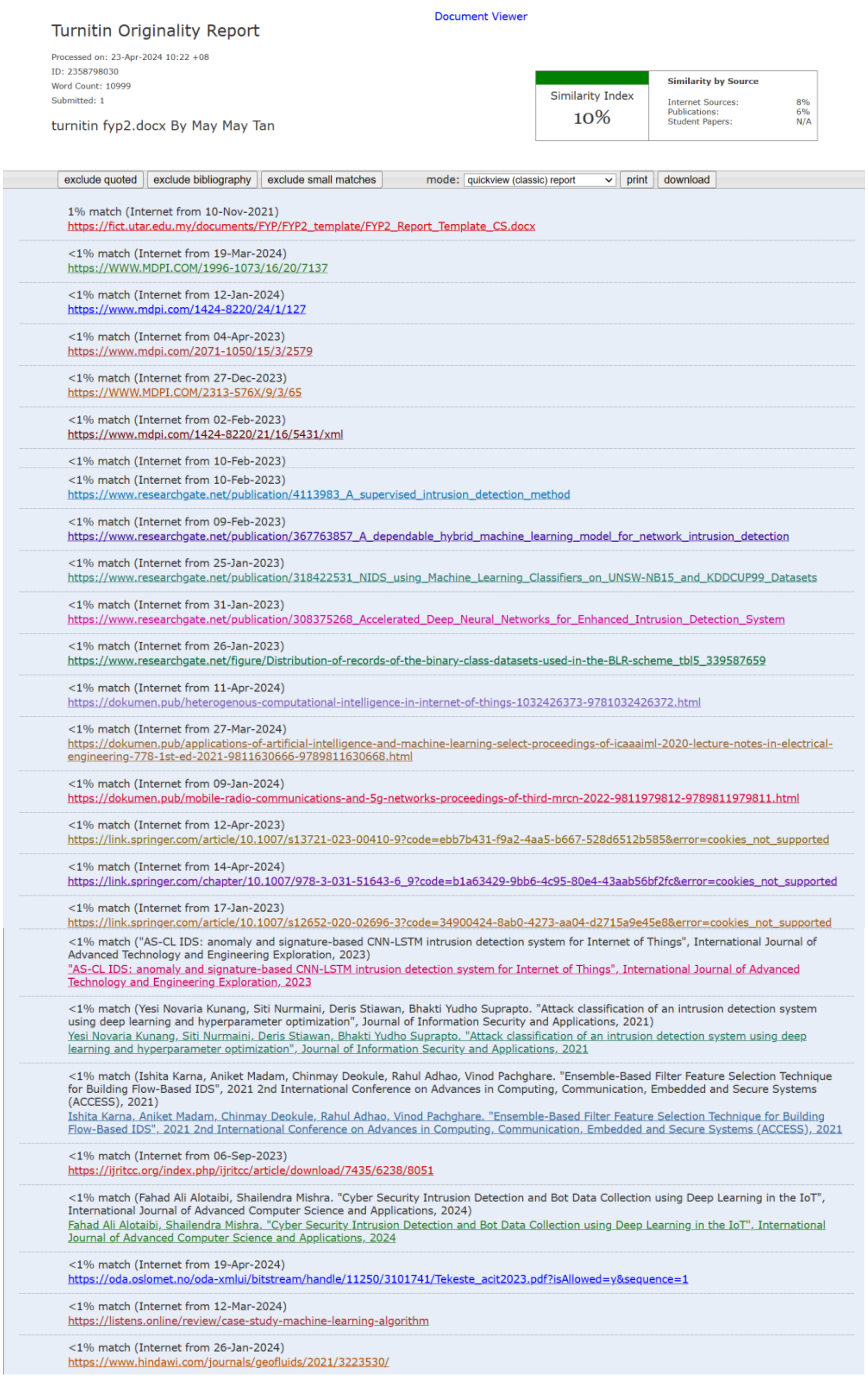
	Accuracy	Precision	Recall	F1 Score
RF	99.92 / 99.83	99.92 / 99.84	99.92 / 99.83	99.92 / 99.84
DNN	99.81 / 99.66	99.81 / 99.72	99.81 / 99.66	99.81 / 99.65
DAE	99.87 / 99.63	99.87 / 99.68	99.87 / 99.63	99.87 / 99.63

Conclusion

- Data pre-processing and feature selection had chosen significant features in dataset.
- Hyperparameter tuning then increased the performance of existing models compared with previous researchers' models.

Research Student: Tan May May
Research Supervisor: Cik Zanariah Binti Zainudin

PLAGIARISM CHECK RESULT



Document Viewer

Turnitin Originality Report

Processed on: 23-Apr-2024 10:22 +08
ID: 2358798030
Word Count: 10999
Submitted: 1

turnitin fyp2.docx By May May Tan

Similarity Index	Similarity by Source	
10%	Internet Sources:	8%
	Publications:	6%
	Student Papers:	N/A

1% match (Internet from 10-Nov-2021)
https://fict.utar.edu.my/documents/FYP/FYP2_template/FYP2_Report_Template_CS.docx

<1% match (Internet from 19-Mar-2024)
https://WWW.MDPI.COM/1996-1073/16/20/7137

<1% match (Internet from 12-Jan-2024)
https://www.mdpi.com/1424-8220/24/1/127

<1% match (Internet from 04-Apr-2023)
https://www.mdpi.com/2071-1050/15/3/2579

<1% match (Internet from 27-Dec-2023)
https://WWW.MDPI.COM/2313-576X/9/3/65

<1% match (Internet from 02-Feb-2023)
https://www.mdpi.com/1424-8220/21/16/5431/xml

<1% match (Internet from 10-Feb-2023)

<1% match (Internet from 10-Feb-2023)
https://www.researchgate.net/publication/4113983_A_supervised_intrusion_detection_method

<1% match (Internet from 09-Feb-2023)
https://www.researchgate.net/publication/367763857_A_dependable_hybrid_machine_learning_model_for_network_intrusion_detection

<1% match (Internet from 25-Jan-2023)
https://www.researchgate.net/publication/318422531_NIDS_using_Machine_Learning_Classifiers_on_UNSW-NB15_and_KDDCUP99_Datasets

<1% match (Internet from 31-Jan-2023)
https://www.researchgate.net/publication/308375268_Accelerated_Deep_Neural_Networks_for_Enhanced_Intrusion_Detection_System

<1% match (Internet from 26-Jan-2023)
https://www.researchgate.net/figure/Distribution-of-records-of-the-binary-class-datasets-used-in-the-BLR-scheme_tbl5_339587659

<1% match (Internet from 11-Apr-2024)
https://dokumen.pub/heterogenous-computational-intelligence-in-internet-of-things-1032426373-9781032426372.html

<1% match (Internet from 27-Mar-2024)
https://dokumen.pub/applications-of-artificial-intelligence-and-machine-learning-select-proceedings-of-icaaaiml-2020-lecture-notes-in-electrical-engineering-778-1st-ed-2021-9811630666-9789811630668.html

<1% match (Internet from 09-Jan-2024)
https://dokumen.pub/mobile-radio-communications-and-5g-networks-proceedings-of-third-mrcn-2022-9811979812-9789811979811.html

<1% match (Internet from 12-Apr-2023)
https://link.springer.com/article/10.1007/s13721-023-00410-9?code=ebb7b431-f9a2-4aa5-b667-528d6512b585&error=cookies_not_supported

<1% match (Internet from 14-Apr-2024)
https://link.springer.com/chapter/10.1007/978-3-031-51643-6_9?code=b1a63429-9bb6-4c95-80e4-43aab56bf2fc&error=cookies_not_supported

<1% match (Internet from 17-Jan-2023)
https://link.springer.com/article/10.1007/s12652-020-02696-3?code=34900424-8ab0-4273-aa04-d2715a9e45e8&error=cookies_not_supported

<1% match ("AS-CL IDS: anomaly and signature-based CNN-LSTM intrusion detection system for Internet of Things", International Journal of Advanced Technology and Engineering Exploration, 2023)
"AS-CL IDS: anomaly and signature-based CNN-LSTM intrusion detection system for Internet of Things", International Journal of Advanced Technology and Engineering Exploration, 2023

<1% match (Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, Bhakti Yudho Suprapto. "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization", Journal of Information Security and Applications, 2021)
Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, Bhakti Yudho Suprapto. "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization", Journal of Information Security and Applications, 2021

<1% match (Ishita Karna, Aniket Madam, Chinmay Deokule, Rahul Adhao, Vinod Pachghare. "Ensemble-Based Filter Feature Selection Technique for Building Flow-Based IDS", 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), 2021)
Ishita Karna, Aniket Madam, Chinmay Deokule, Rahul Adhao, Vinod Pachghare. "Ensemble-Based Filter Feature Selection Technique for Building Flow-Based IDS", 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), 2021

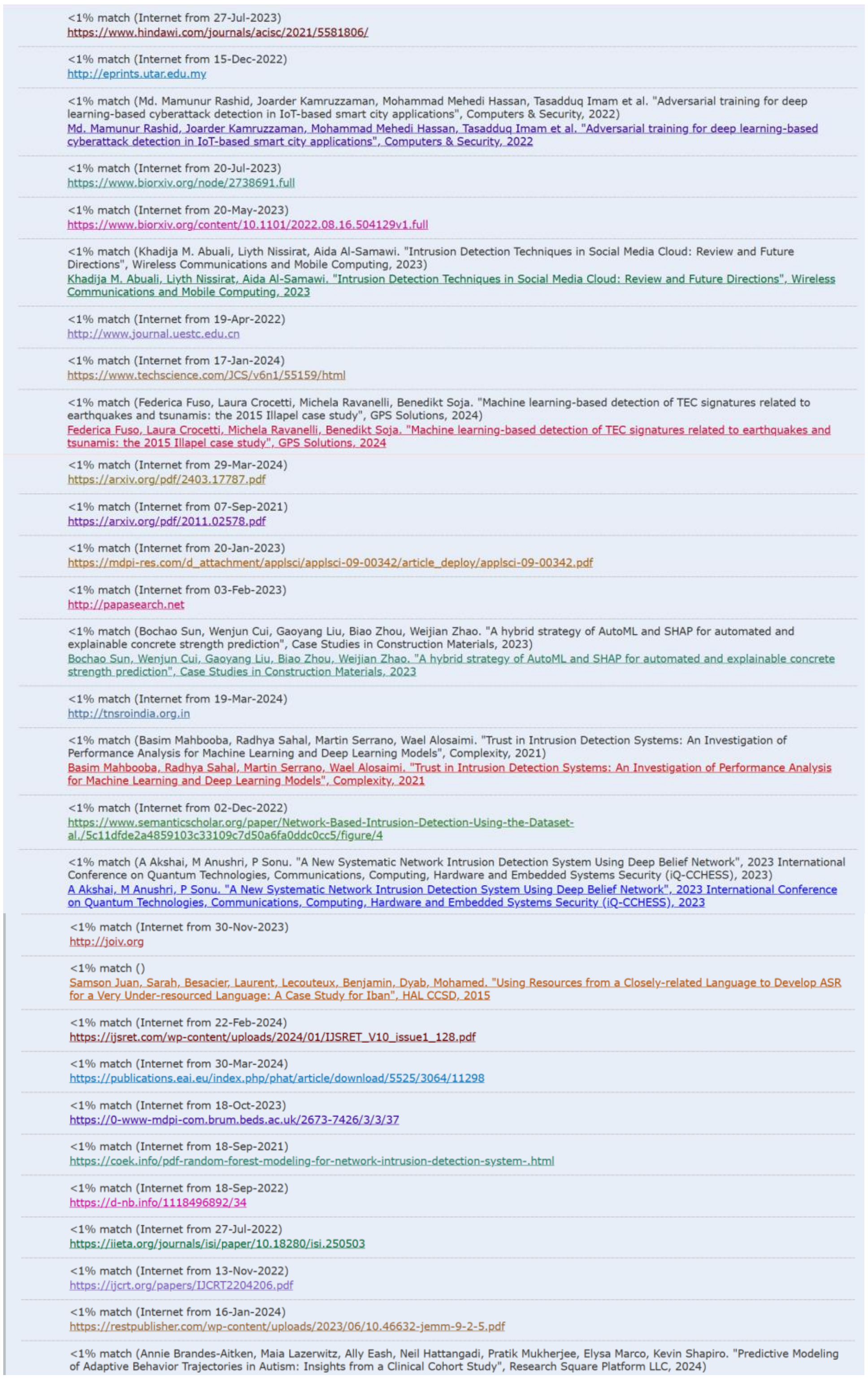
<1% match (Internet from 06-Sep-2023)
https://ijritcc.org/index.php/ijritcc/article/download/7435/6238/8051

<1% match (Fahad Ali Alotaibi, Shailendra Mishra. "Cyber Security Intrusion Detection and Bot Data Collection using Deep Learning in the IoT", International Journal of Advanced Computer Science and Applications, 2024)
Fahad Ali Alotaibi, Shailendra Mishra. "Cyber Security Intrusion Detection and Bot Data Collection using Deep Learning in the IoT", International Journal of Advanced Computer Science and Applications, 2024

<1% match (Internet from 19-Apr-2024)
https://oda.oslomet.no/oda-xmlui/bitstream/handle/11250/3101741/Tekeste_acit2023.pdf?isAllowed=y&sequence=1

<1% match (Internet from 12-Mar-2024)
https://listens.online/review/case-study-machine-learning-algorithm

<1% match (Internet from 26-Jan-2024)
https://www.hindawi.com/journals/geofluids/2021/3223530/


<1% match (Internet from 27-Jul-2023)
https://www.hindawi.com/journals/acisc/2021/5581806/

<1% match (Internet from 15-Dec-2022)
http://eprints.utar.edu.my

<1% match (Md. Mamunur Rashid, Joarder Kamruzzaman, Mohammad Mehedi Hassan, Tasadduq Imam et al. "Adversarial training for deep learning-based cyberattack detection in IoT-based smart city applications", Computers & Security, 2022)
Md. Mamunur Rashid, Joarder Kamruzzaman, Mohammad Mehedi Hassan, Tasadduq Imam et al. "Adversarial training for deep learning-based cyberattack detection in IoT-based smart city applications", Computers & Security, 2022

<1% match (Internet from 20-Jul-2023)
https://www.biorxiv.org/node/2738691.full

<1% match (Internet from 20-May-2023)
https://www.biorxiv.org/content/10.1101/2022.08.16.504129v1.full

<1% match (Khadija M. Abuali, Liyth Nissirat, Aida Al-Samawi. "Intrusion Detection Techniques in Social Media Cloud: Review and Future Directions", Wireless Communications and Mobile Computing, 2023)
Khadija M. Abuali, Liyth Nissirat, Aida Al-Samawi. "Intrusion Detection Techniques in Social Media Cloud: Review and Future Directions", Wireless Communications and Mobile Computing, 2023

<1% match (Internet from 19-Apr-2022)
http://www.journal.uestc.edu.cn

<1% match (Internet from 17-Jan-2024)
https://www.techscience.com/JCS/v6n1/55159/html

<1% match (Federica Fuso, Laura Crocetti, Michela Ravanelli, Benedikt Soja. "Machine learning-based detection of TEC signatures related to earthquakes and tsunamis: the 2015 Illapel case study", GPS Solutions, 2024)
Federica Fuso, Laura Crocetti, Michela Ravanelli, Benedikt Soja. "Machine learning-based detection of TEC signatures related to earthquakes and tsunamis: the 2015 Illapel case study", GPS Solutions, 2024

<1% match (Internet from 29-Mar-2024)
https://arxiv.org/pdf/2403.17787.pdf

<1% match (Internet from 07-Sep-2021)
https://arxiv.org/pdf/2011.02578.pdf

<1% match (Internet from 20-Jan-2023)
https://mdpi-res.com/d_attachment/applsci/applsci-09-00342/article_deploy/applsci-09-00342.pdf

<1% match (Internet from 03-Feb-2023)
http://papasearch.net

<1% match (Bochao Sun, Wenjun Cui, Gaoyang Liu, Biao Zhou, Weijian Zhao. "A hybrid strategy of AutoML and SHAP for automated and explainable concrete strength prediction", Case Studies in Construction Materials, 2023)
Bochao Sun, Wenjun Cui, Gaoyang Liu, Biao Zhou, Weijian Zhao. "A hybrid strategy of AutoML and SHAP for automated and explainable concrete strength prediction", Case Studies in Construction Materials, 2023

<1% match (Internet from 19-Mar-2024)
http://tnsroindia.org.in

<1% match (Basim Mahbooba, Radhya Sahal, Martin Serrano, Wael Alosaimi. "Trust in Intrusion Detection Systems: An Investigation of Performance Analysis for Machine Learning and Deep Learning Models", Complexity, 2021)
Basim Mahbooba, Radhya Sahal, Martin Serrano, Wael Alosaimi. "Trust in Intrusion Detection Systems: An Investigation of Performance Analysis for Machine Learning and Deep Learning Models", Complexity, 2021

<1% match (Internet from 02-Dec-2022)
https://www.semanticscholar.org/paper/Network-Based-Intrusion-Detection-Using-the-Dataset-al./5c11dfde2a4859103c33109c7d50a6fa0ddc0cc5/figure/4

<1% match (A Akshai, M Anushri, P Sonu. "A New Systematic Network Intrusion Detection System Using Deep Belief Network", 2023 International Conference on Quantum Technologies, Communications, Computing, Hardware and Embedded Systems Security (iQ-CCHESS), 2023)
A Akshai, M Anushri, P Sonu. "A New Systematic Network Intrusion Detection System Using Deep Belief Network", 2023 International Conference on Quantum Technologies, Communications, Computing, Hardware and Embedded Systems Security (iQ-CCHESS), 2023

<1% match (Internet from 30-Nov-2023)
http://joiv.org

<1% match ()
Samson Juan, Sarah, Besacier, Laurent, Lecouteux, Benjamin, Dyab, Mohamed. "Using Resources from a Closely-related Language to Develop ASR for a Very Under-resourced Language: A Case Study for Iban", HAL CCSD, 2015

<1% match (Internet from 22-Feb-2024)
https://ijsret.com/wp-content/uploads/2024/01/IJSRET_V10_issue1_128.pdf

<1% match (Internet from 30-Mar-2024)
https://publications.eai.eu/index.php/phat/article/download/5525/3064/11298

<1% match (Internet from 18-Oct-2023)
https://0-www-mdpi-com.brum.beds.ac.uk/2673-7426/3/3/37

<1% match (Internet from 18-Sep-2021)
https://coek.info/pdf-random-forest-modeling-for-network-intrusion-detection-system-.html

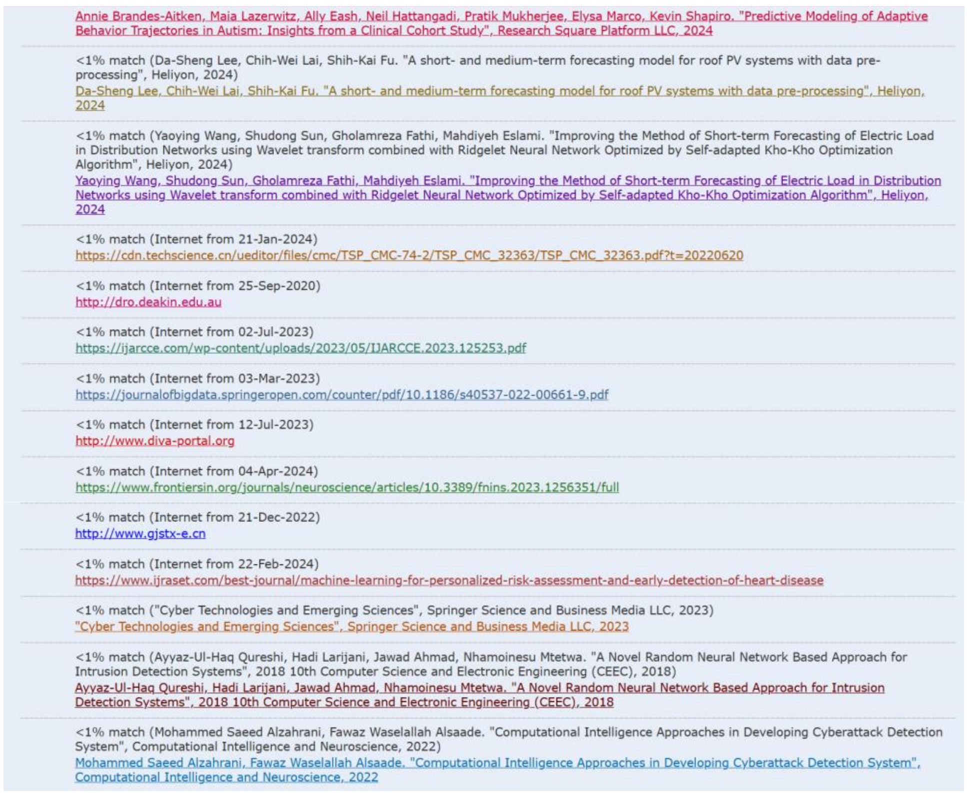
<1% match (Internet from 18-Sep-2022)
https://d-nb.info/1118496892/34

<1% match (Internet from 27-Jul-2022)
https://iieta.org/journals/isi/paper/10.18280/isi.250503

<1% match (Internet from 13-Nov-2022)
https://ijcrt.org/papers/IJCRT2204206.pdf

<1% match (Internet from 16-Jan-2024)
https://restpublisher.com/wp-content/uploads/2023/06/10.46632-jemm-9-2-5.pdf

<1% match (Annie Brandes-Aitken, Maia Lazerwitz, Ally Eash, Neil Hattangadi, Pratik Mukherjee, Elysa Marco, Kevin Shapiro. "Predictive Modeling of Adaptive Behavior Trajectories in Autism: Insights from a Clinical Cohort Study", Research Square Platform LLC, 2024)



Annie Brandes-Aitken, Maia Lazerwitz, Ally Eash, Neil Hattangadi, Pratik Mukherjee, Elysa Marco, Kevin Shapiro. "Predictive Modeling of Adaptive Behavior Trajectories in Autism: Insights from a Clinical Cohort Study", Research Square Platform LLC, 2024

<1% match (Da-Sheng Lee, Chih-Wei Lai, Shih-Kai Fu. "A short- and medium-term forecasting model for roof PV systems with data pre-processing", Heliyon, 2024)
Da-Sheng Lee, Chih-Wei Lai, Shih-Kai Fu. "A short- and medium-term forecasting model for roof PV systems with data pre-processing", Heliyon, 2024

<1% match (Yaoying Wang, Shudong Sun, Gholamreza Fathi, Mahdiyeh Eslami. "Improving the Method of Short-term Forecasting of Electric Load in Distribution Networks using Wavelet transform combined with Ridgelet Neural Network Optimized by Self-adapted Kho-Kho Optimization Algorithm", Heliyon, 2024)
Yaoying Wang, Shudong Sun, Gholamreza Fathi, Mahdiyeh Eslami. "Improving the Method of Short-term Forecasting of Electric Load in Distribution Networks using Wavelet transform combined with Ridgelet Neural Network Optimized by Self-adapted Kho-Kho Optimization Algorithm", Heliyon, 2024

<1% match (Internet from 21-Jan-2024)
https://cdn.techscience.cn/ueditor/files/cmc/TSP_CMC-74-2/TSP_CMC_32363/TSP_CMC_32363.pdf?t=20220620

<1% match (Internet from 25-Sep-2020)
http://dro.deakin.edu.au

<1% match (Internet from 02-Jul-2023)
https://ijarcce.com/wp-content/uploads/2023/05/IJARCCE.2023.125253.pdf

<1% match (Internet from 03-Mar-2023)
https://journalofbigdata.springeropen.com/counter/pdf/10.1186/s40537-022-00661-9.pdf

<1% match (Internet from 12-Jul-2023)
http://www.diva-portal.org

<1% match (Internet from 04-Apr-2024)
https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2023.1256351/full

<1% match (Internet from 21-Dec-2022)
http://www.gjstx-e.cn

<1% match (Internet from 22-Feb-2024)
https://www.ijraset.com/best-journal/machine-learning-for-personalized-risk-assessment-and-early-detection-of-heart-disease

<1% match ("Cyber Technologies and Emerging Sciences", Springer Science and Business Media LLC, 2023)
"Cyber Technologies and Emerging Sciences", Springer Science and Business Media LLC, 2023

<1% match (Ayyaz-Ul-Haq Qureshi, Hadi Larijani, Jawad Ahmad, Nhamoinesu Mtetwa. "A Novel Random Neural Network Based Approach for Intrusion Detection Systems", 2018 10th Computer Science and Electronic Engineering (CEEC), 2018)
Ayyaz-Ul-Haq Qureshi, Hadi Larijani, Jawad Ahmad, Nhamoinesu Mtetwa. "A Novel Random Neural Network Based Approach for Intrusion Detection Systems", 2018 10th Computer Science and Electronic Engineering (CEEC), 2018

<1% match (Mohammed Saeed Alzahrani, Fawaz Waselallah Alsaade. "Computational Intelligence Approaches in Developing Cyberattack Detection System", Computational Intelligence and Neuroscience, 2022)
Mohammed Saeed Alzahrani, Fawaz Waselallah Alsaade. "Computational Intelligence Approaches in Developing Cyberattack Detection System", Computational Intelligence and Neuroscience, 2022

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Tan May May
ID Number(s)	20ACB01863
Programme / Course	Bachelor of Information Technology (Hons) Communications and Networking
Title of Final Year Project	Intrusion Detection System (IDS) using Machine Learning

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 10 % **Similarity by source** Internet Sources: 8 % Publications: 6 % Student Papers: 0 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

____
Signature of Supervisor

Name: Cik Zanariah binti Zainudin

Date: 23/4/24

____
Signature of Co-Supervisor

Name: ______________________

Date: ______________________

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	20ACB01863
Student Name	Tan May May
Supervisor Name	Cik Zanariah binti Zainudin

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

[illegible]
____
(Signature of Student)
Date: 23/4/24