COLLEGE OF COMPUTING AND INFORMATICS

UNIVERSITI TENAGA NASIONAL

NETWORK INTRUSION DETECTION MODEL BASED ON REAL AND ENCRYPTED SYNTHETIC ATTACK TRAFFIC USING DECISION TREE ALGORITHM

MIKHAIL AMZAR BIN KAMARUDDIN

2023

NETWORK INTRUSION DETECTION MODEL BASED ON REAL AND ENCRYPTED SYNTHETIC ATTACK TRAFFIC USING DECISION TREE ALGORITHM

by

MIKHAIL AMZAR BIN KAMARUDDIN

Project Supervisor: Md Nabil Bin Ahmad Zawawi, TS

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE BACHELOR OF COMPUTER SCIENCE, COLLEGE OF COMPUTING AND INFORMATICS
UNIVERSITI TENAGA NASIONAL

2022

DECLARATION

I hereby declare that this report, submitted to University Tenaga Nasional as a partial fulfillment of the requirements for the Bachelor of Computer Science has not been submitted as an exercise for a degree at any other university. I also certify that the work described here is entirely my own except for excerpts and summaries whose sources are appropriately cited in the references.

This report may be made available within the university library and may be photocopied or loaned to other libraries for the purposes of consultation.

[illegible]

2 June 2022

MIKHAIL AMZAR BIN KAMARUDDIN
CS0107477

APPROVAL SHEET

This thesis entitled:

“Network Intrusion Detection Model Based On Real and Encrypted Synthetic Attack Traffic Using Decision Algorithm”

Submitted by:

MIKHAIL AMZAR BIN KAMARUDDIN (CS0107477)

In requirement for the degree of Bachelor of Computer Science, College Of Computing and Informatics, University Tenaga Nasional has been accepted.

Supervisor: Md Nabil Bin Ahmad Zawawi, TS

Signature: ……………………………

Date: 2 June 2022

ABSTRACT

Network intrusion detection systems (NIDS) are essential for defending networks from online security threats. In this research, we try to develop a network intrusion detection model based on decision tree algorithm using HIKARI-21 dataset and observe how effective a decision tree algorithm can be utilized on an unbalanced dataset such as HIKARI-21. Our method makes use of a sizeable dataset of network traffic with labels specifying whether each instance is a non-malicious traffic or a malicious traffic. Using this dataset, the decision tree model will be trained to identify the patterns and traits of both normal and malicious traffic. For this paper, focus is given on developing the model and understanding related topics such as machine learning techniques and methodologies to ease any future work in similar topics using the same dataset. Additionally, exploring numerous possible tools for use in the project to develop the network intrusion detection system. Overall, this study shows the potential of utilizing a decision tree algorithm for a network intrusion detection model and emphasizes the significance of taking both accuracy and efficiency into account when designing such systems.

TABLE OF CONTENTS

ABSTRACT 5
LIST OF TABLES 8
LIST OF FIGURES 9
CHAPTER 1 10
INTRODUCTION 10
  1.1 Background 10
  1.2 Problem Statement 11
  1.3 Objective 12
  1.4 Scope 12
    User Scope 12
    System Scope 12
  1.5 Expected Outcomes 12
CHAPTER 2 13
LITERATURE REVIEW 13
  2.1 Intrusion Detection System 13
  2.2 Machine Learning 15
    2.2.1 Managing Data 17
    2.2.2 Model Training 18
    2.2.3 Model Evaluation 18
    2.2.4 Model Deployment 19
    2.2.5 Decision Tree Algorithm 19
  2.3 Machine Learning Approach to IDS 21
  2.4 Evaluation Metrics for Classification 22
  2.5 Tools 25
    Anaconda Navigator 25
    Google Colab 26
    Weka 27
  2.6 Dashboard 28
    2.6.1 Grafana 28
    2.6.2 Neptune.ai 29

2.6.3 .NET 29
2.7 Software Development Methodology 30
2.7.1 Waterfall Model 30
2.7.2 Agile 32
2.8 Similar System 32
2.8.1 Splunk 33
2.8.2 FireEye 33
CHAPTER 3 35
METHODOLOGY 35
3.1 Chosen Software Development Methodology 35
3.2 Proposed System Features 37
3.3 Chosen Machine Learning Tool 37
3.4 Chosen Dashboard Tool 38
CHAPTER 4 39
DATASET ANALYSIS 39
4.1 HIKARI-2021 39
4.2 Dataset Features and Classes 42
CHAPTER 5 47
IMPLEMENTATION 47
5.1 Data preprocessing 47
5.2 Imbalanced Dataset 47
5.3 Train Test split 48
5.4 Multi-class Classification Results 48
5.4.1 dt_default: Multi-class Classification with imbalanced distribution dataset 48
5.4.2 Comparison with other algorithms. 50
5.5 Binary Classification 51
5.6 Dashboard in Neptune.ai 54
CHAPTER 6 57
CONCLUSION 57
REFERENCES 59

LIST OF TABLES

Table No.		Page
2.1	Intrusion Detection System by implementation	14
2.2	Intrusion Detection System by detection method	15
4.1	HIKARI-21 Content requirements	39
4.2	HIKARI-21 comparison with other IDS datasets	41
4.3	HIKARI-21 features	45
5.1	Score results	48
5.2	Score results comparison	50
5.3	Score results comparison for binary classification of Background against Malicious traffic	51
5.4	Score results comparison for binary classification of Background against Malicious traffic	52
5.5	Code to send model evaluation score to Neptune.ai	55

LIST OF FIGURES

Figure No.		Page
2.1	Machine Learning Modelling Cycle	17
2.1	Decision tree example	20
2.3	Confusion matrix example.	23
2.4	Anaconda interface and logo	26
2.5	Google Colab logo and interface	27
2.6	Waterfall model diagram	30
4.1	HIKARI-21 target class (traffic_category) distribution	43
5.1	dt_default classes score results	49
5.2	dt_default confusion matrix	50
5.3	Establishing connection to Neptune.ai from Google Colab	55
5.4	Neptune.ai dashboard page	56
5.5	Neptune.ai evaluation score log	56

CHAPTER 1

INTRODUCTION

1.1 Background

The application of machine learning to cybersecurity and network threat detection is a rapidly growing field of study, with numerous new methods and software designed to enhance the effectiveness of these systems. Consequently, it is anticipated that machine learning will play a significant role in the future improvement of the security of computer systems and networks. In recent years, the growing number of cyberattacks has rendered traditional rule-based and signature-based security systems incapable of detecting and preventing these threats. This has increased interest in machine learning as a means for developing more intelligent, adaptive, and automated security systems that can derive information from data and identify threats not previously identified. Various aspects of cybersecurity and threat detection, such as intrusion detection, malware detection, anomaly detection, and phishing detection, can be approached with machine learning algorithms and methodologies. By assessing network traffic and email, for instance, machine learning algorithms can detect patterns that indicate a possible threat and prevent attacks in advance. In this paper, focus is given on developing a machine learning model using decision tree algorithm to detect attack traffic in a network flow dataset, essentially, a network intrusion detection model. Using Google Colab as a platform to write Python codes and utilizing scikit-learn and other useful packages to conduct the testing. The dataset that will be used for the classification model is HIKARI-21.

1.2 Problem Statement

One of the primary concerns surrounding network is security. With the world now being so connected to the internet, naturally there will be increased total of malicious activity committed on networks. One of these malicious activities is a network intrusion. The act of gaining illegal access or attempting malicious action to enter a computer network or system, is known as network intrusion. It can be carried out by a single person or by a group, and it can be driven by a wide range of purposes, including the theft of sensitive data, the disruption of the operation of the network, or the gaining of illegal access to resources. It is essential to employ security measures such as firewalls, antivirus software, and frequent security upgrades in order to prevent unauthorized access to a computer network. It is also essential to educate users about the dangers posed by network intrusions, as well as the ways in which such attacks can be recognized and avoided Security systems such as a network intrusion detection system have been developed for many years now. Network intrusion detection system monitors and analyzes network traffic and alerts the responsible party if any anomaly or suspicious activity are detected. The goal for this project is to develop a network intrusion detection model that will help us achieve such results using datasets of real and encrypted synthetic attack traffic while utilizing a decision tree algorithm and will provide user a dashboard containing relevant results concerning data of the detection.

1.3 Objective

The project objectives are:

i. To design a network intrusion detection model that applies Decision Tree algorithm.
ii. To produce a machine learning model and dashboard for network intrusion detection model.

1.4 Scope

User Scope

Users using the model and dashboard to observe metrics of the model and its related details.

System Scope

The scope of the system consists of implementing decision tree machine learning algorithm. Additionally, the system will only perform intrusion detection. Dataset that will be used for the training of the algorithm is a publicly available dataset – ALLFLOWMETER_HIKARI2021.csv.

1.5 Expected Outcomes

i. A machine learning model and dashboard must be produced for the network intrusion system.
ii. The network intrusion detection model should apply Decision Tree algorithm.

CHAPTER 2

LITERATURE REVIEW

Literature review consist of reading various research regarding the implementation of machine learning in an intrusion detection system to better understand the techniques and approaches taken when dealing with classification model for an intrusion detection system.

2.1 Intrusion Detection System

An Intrusion Detection System (IDS) is a device or software that monitors a network or system for malicious activity or violations of policies. Intrusion detection and prevention systems are primarily concerned with identifying potential intrusions in systems and networks, logging data, reporting attempts, and learning. Currently, machine learning (ML) technologies have been implemented to enhance the effectiveness of intrusion detection systems, which are one of the most widely used security infrastructures to defend networks against attacks [6]. Network Intrusion Detection System is capable of being configured to monitor traffic on a particular network segment or over the entirety of the network. To identify potentially malicious behavior, it employs a number of different methods, including signature-based detection and anomaly-based detection. In signature-based detection, established harmful traffic patterns are looked for. This works well for identifying known attacks, but it might miss new or unusual attack types. On the other hand, Anomaly-based detection refers to detecting anomalies based on deviations from typical network behavior. This can be useful for spotting novel attack vectors, but it may also identify legitimate traffic as suspicious, leading to a lot of false positives.

Table 2.1: Intrusion Detection System by implementation

Type	Implementation
Network-based IDS (NIDS)	Placed in strategic points within the network, typically data chokepoints.
Host-based IDS (HIDS)	Runs on the host system it is placed in.

Table 2.2: Intrusion Detection System by detection method

Detection model	Description
Signature-based IDS (SIDS)	Examines network packets and compares them to a database of known attack signatures or features. This type of IDS searches for specified patterns, such as byte or instruction sequences.
Anomaly-based IDS (AIDS)	Detects current network traffic and compares trends to a baseline. It detects malicious activity patterns rather than specific data patterns, going beyond the

	attack signature approach.

2.2 Machine Learning

The development of algorithms and statistical models that enable computers to learn from data and make judgements based on it without being explicitly programmed is known as machine learning, and it is a subfield of artificial intelligence. In the field of machine learning, an algorithm is taught to make predictions or choices by being exposed to a dataset during its training phase. For example, a machine learning algorithm might be trained on a collection of consumer data for an online shopping platform, including information about their search history, purchases, and cart contents. The algorithm might therefore be used to anticipate which items the customer might buy next. Machine learning comes in a variety of kinds, including reinforcement learning, unsupervised learning, semi-supervised learning, and supervised learning.

Table 2.3: Types of machine learning methods

Machine Learning Type	Description
Supervised learning	A training dataset is needed for supervised learning, which must include labelled responses or output targets as well as examples for the input. The ML model are then calibrated using the pairs of input and

	output data from the training set. Following a model's successful training, it can be used to forecast the target output using fresh or previously unobserved data points of the input attributes.
Unsupervised learning	Learning system is expected to identify patterns without any labels or guidelines in place, this is known as unsupervised learning. Training data only consists of variables x with the goal of finding structural information of interest. For example, grouping data of similar attributes or also known as clustering.
Reinforcement learning	We define the system's current state, establish a goal, provide a list of permissible actions and the environmental constraints on their results, and then let the ML model experiment with the process of reaching the goal on its own using the concept of trial and error to maximize a reward.

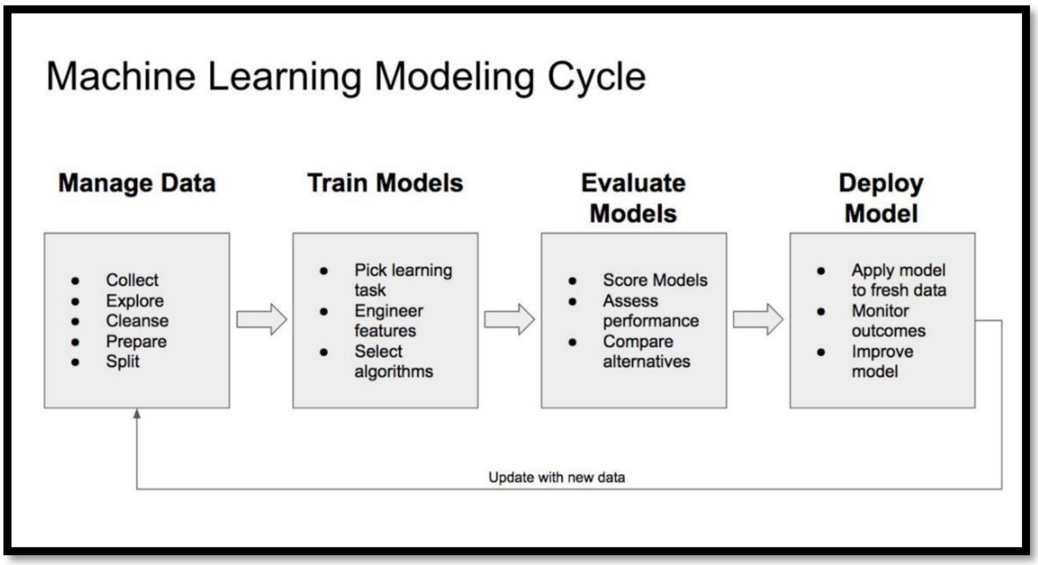


Figure 2.1: Machine Learning Modelling Cycle

2.2.1 Managing Data

The overall machine learning modelling process is a lengthy set of steps that will determine the overall success of the machine learning project. The data that will be used as input for the machine learning algorithm must be prioritized first. Data can be obtained and collected from a variety of sources, including purchasing from vendors, generating synthetic data, open-source datasets, and so on. The datasets should be collected in accordance with the needs of the machine learning project, so careful consideration and research are advised when deciding the data to use. After that, exploring the data which is about analyzing the dataset, identifying any error and values that needs to be labelled and corrected. Following collection and exploration, the dataset must be cleaned because datasets frequently have missing values, labelling errors, and so on. Uncleaned datasets may have an impact on the performance and results of the machine learning model, resulting in a poor machine learning outcome. Cleaning data can be time-consuming, but it is necessary to ensure that the dataset is valid, accurate, complete, consistent, and uniform. Furthermore, a high-quality dataset will greatly improve and accelerate the training process. Following that, feature selection and feature engineering techniques are required to facilitate the process of organizing relevant data for the machine learning model. There are numerous tools available to help speed up and automate the data preparation process. The datasets must then be divided into groups, with each group serving a different purpose. A dataset should essentially be divided into a training set and a test set. This is to evaluate the machine learning model's performance when making predictions based on the dataset. Train dataset is used to train the machine learning model, whereas test dataset is used to evaluate the trained machine learning model.

2.2.2 Model Training

Training the machine learning model consist of several important procedures that needs to be executed correctly to satisfy the requirements and objective of the machine learning model. But first, it is important to choose the type of machine learning model that is suitable for the intended task of the project. Starting with the type of learning task, a machine learning task is a form of prediction or inference made based on the problem or query and the available data. The classification task, for example, allocates data to categories, whereas the clustering task groups data based on similarity.

2.2.3 Model Evaluation

Evaluating the machine learning model is essential after training the model to determine if the performance is meeting expectation or not. Additionally, evaluation is important because we need to know whether the machine learning model can make accurate predictions. This phase involves providing the test dataset to the machine learning model. There are several metrics to consider when evaluating models such as classification metrics and regression metrics.

2.2.4 Model Deployment

This step is where the deployment of the finished machine learning model takes place. The process of putting a fully functional machine learning model into production so that it can make data-driven predictions. These predictions are then used by users, developers, and systems to make real business decisions. However, these models are still being monitored to see if any changes occur in the models. A machine learning model might degrade over time for a variety of reasons. Therefore, machine learning model can undergo improvement by training it with new data.

2.2.5 Decision Tree Algorithm

In the field of machine learning, one type of algorithm that is used for classification and regression tasks is called a decision tree. It is a tree-like structure like a flowchart that displays a collection of decisions and the probable outcomes of those actions. The "decisions that need to be made" are represented by the "nodes" in the tree, and the "potential outcomes" of those decisions are represented by the "edges" in the tree. Decision trees are a method for modelling decisions and outcomes, which map decisions using a branching structure. Decision trees are utilized to determine the likelihood of success for various sequence of decisions made to attain a given objective. The concept of a decision tree predates machine learning, as it may be used to manually model operational decisions in the manner of a flowchart. As a technique for analyzing organizational decision making, they are extensively taught and utilized in business, economics, and operations management. In the context of machine learning, decision trees are a type of supervised learning, this means it train models with tagged input and output datasets. The approach is mostly used to handle classification problems, which involve categorizing or classifying an object using a model. Decision trees are also utilized in regression problems where it is used in predictive analytics to forecast outputs from unknown input [21][22].

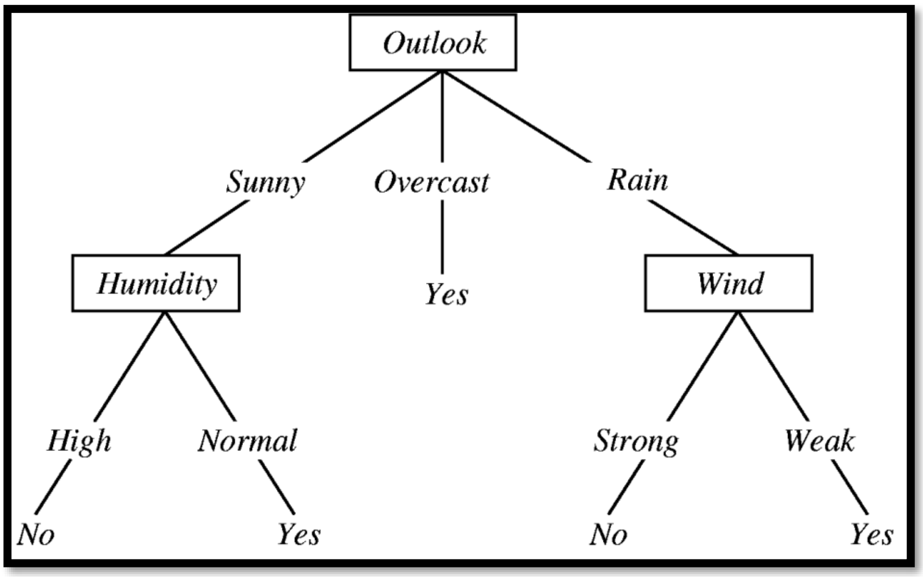


Figure 2.2: Decision tree example

The most typical application of decision trees in machine learning is for classification problems. It is a supervised machine learning issue in which the model is taught to determine whether the data belongs to a known object class or not. Models are trained to label processed data with class labels. In the training phase of the machine learning model lifecycle, the model processes labelled training data to learn the classes. A model needs to comprehend the characteristics that classify a datapoint into the various class labels to solve a classification problem. A classification issue might arise in a variety of contexts. Document classification, image recognition software, and email spam detection are a few examples.

2.3 Machine Learning Approach to IDS

In [6], the author decides to use the CICIDS2017 dataset which contains 14 different attack category and a benign category. The author conducted a binary classification and weighted binary classification to classify whether the class is benign or malignant. Various algorithms are trained and tested, which shows tree-based algorithms like Decision Tree (*Accuracy 99.87%, F-score 99.7%*) and Random Forest (*Accuracy 99.89%, F-score 99.74%*) giving slightly better score in terms of accuracy and F-score, followed by Gradient Boosting (*Accuracy 99.67%, F-score 99.2%)*, Neural Network (*Accuracy 99.5%, F-score 98.7%*), and Logistic Regression (*Accuracy 93.5%, F-score 83.4%*). Tree-based algorithm seems to perform well for a binary classification task. The author also notes that some machine learning algorithms can be used to provide information on which features are important. This can lead to using only the important features for the model which may save training time and achieve better model results. In [3], it is stated that accuracy and computational cost can be better if feature extraction methods are implemented in data pre-processing step. This highlights the importance of selecting the most important and valuable features from the dataset for the model to achieve positive improvements when developing a classification model.

2.4 Evaluation Metrics for Classification

Evaluation metrics are used to evaluate the performance of a machine learning model on a particular task. These metrics provide a quantitative method for evaluating the performance of a model and for comparing the performance of various models. The evaluation metric chosen depends on the problem being solved and the project's objectives.

i. Accuracy

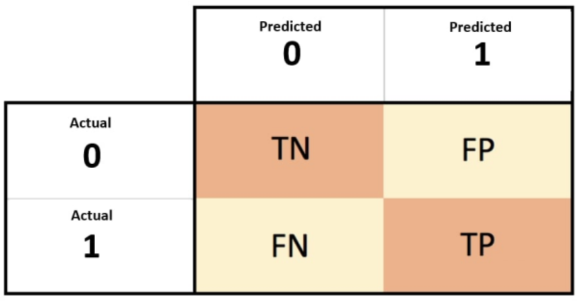
Accuracy is the frequency with which the classifier makes accurate predictions. Accuracy can be defined as the proportion of correct predictions to the total number of predictions.

Accuracy Score = (TP + TN)/ (TP + FN + TN + FP)

It is crucial to remember that a high accuracy rate may not always guarantee that the model is performing well. Accuracy is advantageous when the target class in the dataset is well balanced, but it is not a good choice when the target class is unbalanced [9][11][18]. Additionally, classification accuracy is determined by a simple error count, it would be beneficial to also consider other evaluation metrics.

ii. Confusion Matrix

Confusion Matrix is a performance metric in the form of a table for classification problems in machine learning. It is a summary of prediction results on a classification problem. An example of a confusion matrix is shown in **Figure 2.3**.



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 2.3: Confusion matrix example.

a) True positive (TP)

A test outcome that accurately identifies the existence of a condition or trait.

b) True negative (TN)

A test outcome that correctly identifies the absence of a condition or trait.

c) False positive (FP)

A test outcome which incorrectly indicates that a particular condition or attribute is present.

d) False negative (FN)

A test outcome which incorrectly indicates that a particular condition or attribute is absent.

iii. Precision

The model precision score quantifies the proportion of accurately predicted positive labels out of all positive predictions made. Class distribution influences precision. If there are a greater number of samples from the minority class, then precision will be reduced. Precision can be viewed as an indicator of exactness or quality. To minimize false negatives, we would select a model with a high degree of precision. If we wish to reduce false positives, we will select a model with a high recall [7].

Precision Score = TP / (TruePositive+FalsePositive)

iv. **Recall**

Recall represents the model's ability to accurately predict true positives. In simpler terms, from the total actual positive label, how many are correctly predicted as positive. This contrasts with precision, which measures how many predictions of all the positive predictions made by models are actually positive.

Recall Score = TruePositive / (TruePositive+FalseNegative)

v. **F1 Score**

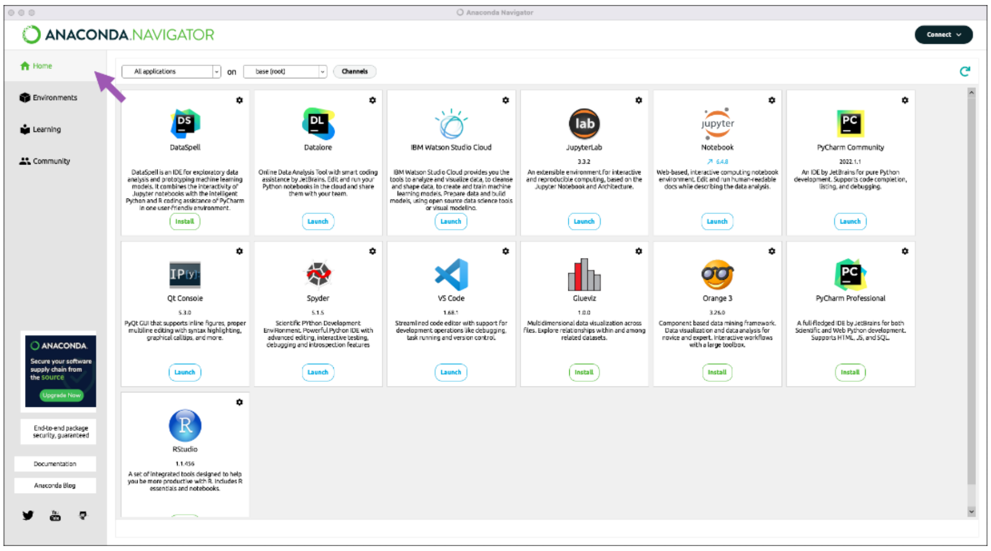
The F1 score represents the model's score as a function of its recall and precision scores. It is an alternative to Accuracy metrics (it does not require us to know the total number of observations). It can be represented as a harmonic mean of precision and recall score.

F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score/)

2.5 Tools

Anaconda Navigator

Anaconda Navigator is a graphical user interface (GUI) that allows users to launch applications and manage packages in the Anaconda Python distribution. It is included with the Anaconda distribution and is designed to make it easier for users to work with Python packages and environments. The GUI can be used to launch Python programs, and operate the conda packages, environments, and channels without using the command line interface commands. Anaconda Navigator is included in the installation of Anaconda Distribution. Anaconda Distribution consist of a wide collection of Python libraries and conda, which is a tool for managing packages. Apart from the numerous open-source packages already available in Anaconda, user can also install thousands of packages from the Anaconda repository and community hub. Anaconda Distribution essentially simplifies the management of Python versions on a single computer and includes a wide number of highly optimized, regularly used data science modules to help users get started quickly. This makes it easier to operate and fulfil scientific packages requirements of Python versions. This software is available to be installed for macOS 10.10 and newer, Windows 10 x86_64 and newer, and Linux with glibc 2.17 or newer. **Figure 2.4** shows the Anaconda user interface.



DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab

3.3.2

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook

↗ 6.4.8

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyCharm Community

2022.1.1

An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.

Qt Console

5.3.0

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder

5.1.5

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code

1.68.1

Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3

3.26.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

PyCharm Professional

A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

RStudio

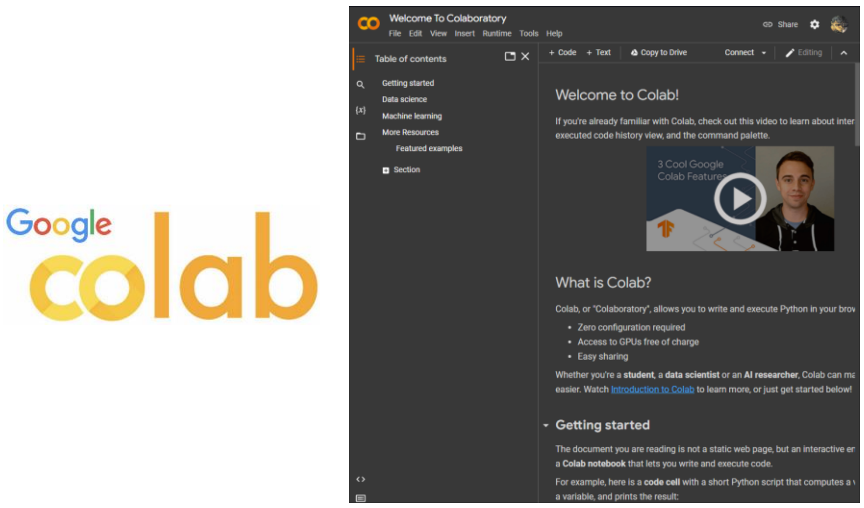
1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 2.4: Anaconda interface and logo.

Google Colab

Google Colab is a product of Google Research. It enables anyone to create and execute arbitrary Python code through the browser. Codes can be executed inside a virtual machine dedicated and private to the user's account. A great advantage of Colab is that it offers free access to computing resources while requiring no setup to use. Colab can be a great tool for the development of machine learning and data science related projects. Google Colab essentially help users write, test, and execute codes in a practical interactive document called notebooks that features cells which allow the creation of space that user can use to insert codes, texts, and other documentative elements. However, it is important to keep in mind that Colab does not provide unlimited computing resources which means it has the possibility to fluctuate due to usage limit. Computing resources are limited due to various reasons. Colab aim to have flexibility in adjusting usage limited and hardware availability because these are the free version of their service. But if the need to access better and guaranteed resources arises, there is Colab Pro, the paid version for Google Colab, available for subscription by users that are packed with many practical and useful upgrades and advantages over the free version. Colab Pro promises powerful GPUS, larger memory capacity, and 100 compute units. Going further up the tier is Colab Pro+ which offer even better resources and benefits.



Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about inter executed code history view, and the command palette.

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your brov

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can ma easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive en a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a v a variable, and prints the result:

Figure 2.5: Google Colab logo and interface.

Weka

Weka is a machine learning software that was developed at the University of Waikato in New Zealand. It is available for free and is open source. It offers a collection of algorithms that can be used for various data mining tasks, such as classification, regression, clustering, and learning to associate rules with categories. Weka also features a graphical user interface that gives users the ability to load data, preprocess that data, and apply machine learning algorithms to that data without having to write any code. In general, Weka is a helpful tool for anyone who is interested in machine learning and data mining. It is particularly well-suited for researching and experimenting with a variety of various algorithms and approaches.

2.6 Dashboard

Analytical dashboard provides insights and help display relevant data to be understood easier by humans. By providing an overview of relevant and impactful data to the user. Through the data displayed on a dashboard, user can recognize trends, statistics, and events to make better decisions and analysis. Dashboards are useful when large and broad complex categorized information requires visualization to execute an accurate analysis of created data.

2.6.1 Grafana

Grafana is a free and open-source data visualization and monitoring tool that enables the construction of dashboards to monitor multiple metrics and data sources. It is typically used for visualizing time series data for monitoring infrastructure and application performance, but it may also be used to visualize data from databases and log files. InfluxDB, Prometheus, and Graphite are among the wide range of data sources that Grafana offer support to. It is equipped with great variety of visualization options, alongside support for graphs, tables, heat maps, and more. Grafana's ability to generate dynamic and configurable dashboards is one of its primary advantages. Dashboards can be created by dragging and dropping panels onto a layout and setting the panel to display the data. It is possible to alter the data presented in each panel using Grafana's query editor. Additionally, Grafana has alerting and notification functionalities in addition to visualization and dashboard capabilities, which help to set up alerts based on thresholds or other situations. When an alert is triggered, Grafana can be configured to send notifications via email, Slack, or other messaging platforms.

2.6.2 Neptune.ai

Neptune.ai is an online platform that is used to log experiment results and model performance metrics regarding machine learning projects. The web application, (app.neptune.ai) can be used for visualization, comparison, monitoring, and collaboration of machine learning projects. It works by connecting the Neptune client to the machine learning environment by writing lines of codes in the same machine learning environment. Every time the model code is executed, the Neptune code should be executed afterwards to establish connection and create a runtime. During this runtime, scoring metrics, images, graphs, and any other required metadata is assigned to a variable and transmitted to the Neptune platform. The logs are stored in our Neptune.ai account and categorized according to groups like runtimes or models category.

2.6.3 .NET

Microsoft created the software development platform known as .NET. for Windows, Linux, and macOS. It offers a foundation for creating, distributing, and operating applications and services. The.NET Standard Library, a collection of libraries that make up .NET, offers numerous features, including support for networking, data access, and file input/output. The Common Language Runtime (CLR), a runtime environment that executes.NET code and controls the resources of the computer on which the code is running, is also included. C#, F#, and VB.NET are just a few of the programming languages that are supported by .NET, giving developers the option to use the one that best suits their needs. It also offers frameworks for creating web, mobile, desktop, and cloud-based apps as well as tools for designing, debugging, and deploying applications. In general, .NET is a strong platform that makes it easier to create and deploy apps and services, and it is utilized by many developers worldwide.

2.7 Software Development Methodology

Software development methodology is the steps of process involved when developing software and the philosophy involved in the completion of the project. Methodology is important when executing projects as it lays out the path, timeline, objectives that help us stay on track during development process.

2.7.1 Waterfall Model

This model is linear and sequential. Each phase must be completed before proceeding with the next phase to ensure organized workflow and no overlapping occur. Consider the image below to understand the Waterfall Model:

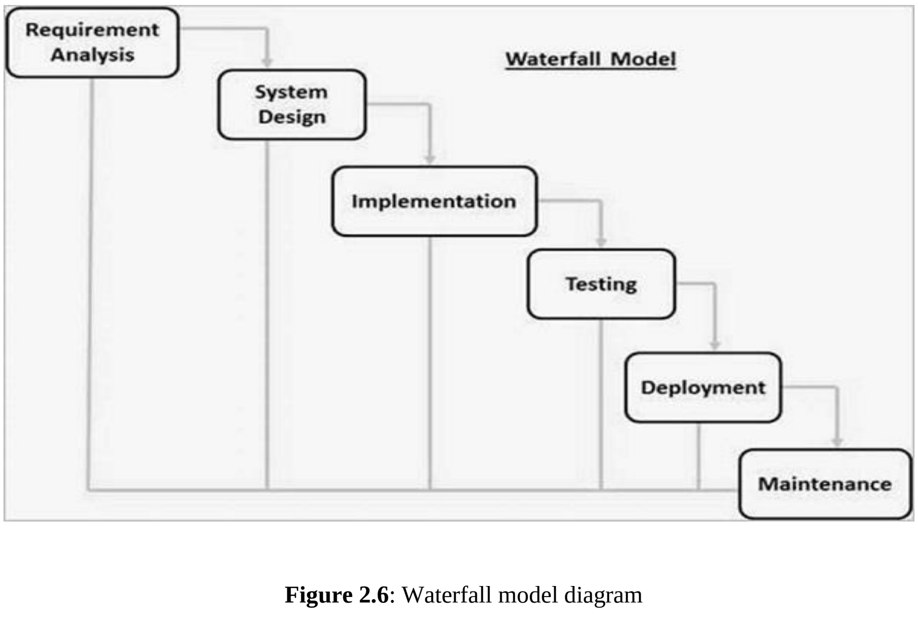


Figure 2.6: Waterfall model diagram

i. **Requirement Gathering and analysis**

During this phase, all potential system requirements are identified and recorded in a specification document.

ii. **System design**

This phase studies the specifications from the previous phase hence initiating system design. This system design aids in determining hardware and system requirements, as well as the overall system architecture.

iii. **Implementation**

The system is first built-in discrete programs called units, with input from the system design phase, and then combined in the following phase. Unit undergo unit testing to test for functioning after it is developed.

iv. **Integration and testing**

Following unit testing, all units generated during the implementation phase are integrated into a system. Following integration, the complete system is tested for flaws and failures.

v. **Deployment**

After the completion of functional and non-functional testing, the product is deployed in the client environment or released to the market.

vi. **Maintenance**

There may be problems or bugs that arise in the client environment. Patches are published to address these vulnerabilities. To improve the product, newer versions are published. Maintenance is performed to implement these modifications in the released software.

Waterfall models are suitable to be applied to projects where the requirements are determined, product definition is stable, expected timeline is relatively short, and the resources and expertise are available.

2.7.2 Agile

Software development using the agile methodology is flexible, iterative, and places a focus on quick prototyping, frequent releases, and ongoing improvement. It is predicated on the Agile Manifesto, a set of guidelines that emphasises customer happiness, functional software, and teamwork and communication. Agile approaches place a heavy emphasis on teamwork and communication among members and are designed to be flexible and responsive to shifting requirements and priorities. Teams are able to quickly react to new knowledge and changes in the business environment as a result, and they are able to provide value to customers fast and effectively. To aid teams in planning, monitoring, and managing their work, agile procedures are frequently used in conjunction with agile project management tools and techniques, such as burndown charts and agile boards. Scrum, Lean, and Extreme Programming are a few of the various agile approaches that have been created over time (XP). Although each of these methodologies has unique procedures and methods, they all follow the guidelines of the Agile Manifesto.

2.8 Similar System

In terms of concept and functionality, there are similar systems such as Splunk and FireEye.

2.8.1 Splunk

Splunk is a software platform that searches, analyses, and visualizes machine-generated data collected from IT infrastructure and business's websites, applications, sensors, and devices. Analyzing these data allow organizations to come up with solutions related to certain aspects of their business. It also provides better understanding of the customer and the service they provide. The data provide an opportunity to be aware of issues regarding their system as well as improving the quality of service or functionality of systems. Machine data are typically complex because it may be presented in an unstructured format which means analyzing it becomes a challenge. Splunk works in a way that it essentially extracts machine data for users in a readable form. With these data user can monitor and record system performance to search for any failure conditions, investigate any occurring events, checking the business matrix, visualizing data to display results and dashboards, and storing data for future reference. Additionally, the benefits of Splunk are why many organizations use this software. Searching for data in Splunk is easy because of the Search Processing Language and the ability to input search data in any form. Splunk also does not require a backend database system due to it using its own file system to store data.

2.8.2 FireEye

FireEye Network Security is a cyber threat solution. Aimed at helping organizations manage risk of breaches. It does this by identifying and halting advanced, targeted, and other evasive attacks hidden in Internet traffic. With real proof, actionable intelligence, and response process integration, it enables quick resolution of reported security events in minutes. FireEye are capable of achieving wide range of detection, prevention, and response measures. Featuring Multi-Vector execution engine, or MVX as they call it and numerous machine learning, AI and correlation engines. FireEye halts the infection of the cyber-attack by detecting previously unseen exploits. Additionally, rule-based analysis based on real-time insights gained on the front lines from thousands of hours of incident response experience, makes it possible for FireEye to identify and stop obfuscated, targeted, and other tailored assaults. FireEye Network Security alerts offer actual, tangible evidence that may be used to respond to targeted and recently identified attacks. For contextual support, threats can also be mapped to the MITRE ATT&CK framework.

CHAPTER 3

METHODOLOGY

3.1 Chosen Software Development Methodology

The chosen methodology is Agile. A flexible and efficient workflow should be the priority when deciding the methodology to be adopted for the project. Agile is the preferred choice as it emphasizes flexibility and rapid iteration. It is intended for complicated and rapidly changing projects, and it is especially well-suited for projects with a high level of uncertainty or ambiguity. Agile enables rapid iteration and delivery: Agile divides work into tiny, focused iterations called "sprints," which typically run two to four weeks. This enables quick deployment of working software and receive feedback from clients early in the process. Any improvement or required changes can be identified and applied after reviewing the results of each sprints. Apart from that, Agile values collaboration, transparency, and communication among developers. This can lead to more informed decisions and a more transparent approach. It would be very beneficial to the development as the deliverables and outcomes can be discussed with the project supervisor throughout the project. Agile is more flexible than the waterfall model, which is more rigid and sequential, because it is built to handle project that might experience multiple changes in design and requirements. Developer can thus respond swiftly and apply changes according to the project's needs. Since Agile enable rapid delivery and continual improvement, it can be less expensive and time-consuming compared to the waterfall model. Typically, in a waterfall model, no finished product can be delivered until the project is nearing the end of its development cycle, which is during the deployment of the system. The Agile methodology will not constraint the progress and work deliverables of the project into strict phases as seen in a waterfall model as well. In a waterfall model, the linear and sequential flow of development means that a phase must be completed first before moving on to work on the next development phase. This element restricts the development of the software in some way as the development of this intrusion detection system comprises of many technical and complicated aspect such as programming, machine learning algorithm implementation, dashboards, and datasets to manage. It is believed that a complicated project would be easier to manage if a methodology like Agile is adopted. Scrum, an implementation of Agile would allow the tasks to be divided to smaller chunks or sprints. When tasks are broken into smaller units, more focus can be put to complete the tasks. Moreover, in an Agile development, a small product or deliverable can be consistently put out after each sprint. This product can be evaluated so that any improvement or change in requirements can be identified to allow applying necessary changes in the product. Division of task should also focus on the features of the system. That way the features can be easily reassessed and improved if needed. Furthermore, Kanban may be used to visualize the workflow of the entire development cycle. A Kanban board should be created to list the task tasks to be done, tasks in progress, and finished task. Work in progress limits is set for the board to avoid having too much task to keep track of.

3.2 Proposed System Features

The network intrusion detection model based on real and encrypted synthetic attack traffic using decision tree algorithm offer a few primary features that outlines the model's primary operations. Firstly, the capability of the model to analyze traffic data in the dataset by recognizing unusual patterns or features. This allows the model to predict the test data and categorize it to the corresponding target class. This leads to an output that displays the model’s performance such as an evaluation metric. The second feature is a dashboard that provides the functionality of logging the evaluation metrics of the models and displaying the outputs back to the user in a form that is organized, practical, and easy to comprehend. A key function of the dashboard is to make it easier for users to observe the model’s overall performance. Additionally, it offers a better user experience, which should enhance user performance when utilizing the model to analyze data and make decisions.

3.3 Chosen Machine Learning Tool

When developing a network intrusion detection model that implements machine learning, a tool that is both powerful and efficient in order to simplify the process of developing machine learning models and Python codes is required. Therefore, the tool for machine learning that will be used is going to be Google Colab. Seamless Python documentation and coding in a single document are available in an interactive environment through Google Colab, which makes it convenient and practical to test codes. The integration of Google Colab with Google Drive offer support to import data from the Drive as well as saving Colab notebooks into the Drive to make sure progress can be saved quickly. Importing Python libraries into Google Colab requires only a few lines of code and does not require the libraries to be downloaded in advance. This makes it possible to import Python packages that are widely used in machine learning projects. Furthermore, Google Colab enables users to share notebooks with one another and work together on those notebooks. This can be especially helpful when working on projects as part of a group or when seeking input from other people in the same field. Most importantly, access to sophisticated hardware for running machine learning models is available through Google Colab and TensorFlow included. This hardware includes GPUs and TPUs, both of which are able to considerably speed up the training process.

3.4 Chosen Dashboard Tool

This network intrusion detection model comes equipped with a dashboard. The user of the network intrusion detection model should be able to make analytical observations and decisions regarding the machine learning model with the help of the dashboard which facilitates the visualization and logging of data and results. Neptune.ai is chosen for its capabilities to conveniently log and store evaluation score and information from the models through an easy and simple integration.

CHAPTER 4

DATASET ANALYSIS

4.1 HIKARI-2021

A dataset is a crucial component of a machine learning model since it contains the data from which the model will learn. The dataset will be used by the model to uncover patterns and relationships in the data, which will then be used to generate predictions. The quality of the dataset is an important consideration when picking which one to use. The quality of the data in the dataset is key to the machine learning model's performance. The model's predictions or conclusions may be unreliable if the data is incomplete, imprecise, or biased. For this network intrusion detection model, HIKARI-2021 was chosen as the dataset to train the machine learning model. HIKARI-2021 was created using a combination of ground-truth data and data that was missing from previous existing IDS datasets. **Table 4.1** shows the content requirements determined for the development of the dataset which emphasizes the required information to be contained in the dataset.

Table 4.1: HIKARI-21 Content requirements.

Content Requirements	Description

Complete capture	Complete capture of the network traffic which refers to communication between host, broadcast message, domain lookup query, the protocol being used. Both flow data and pcap should be available as well.
Payload	A payload is not required for a flow-based method. But producing comprehensive information and making the most of the data are essential. HIKARI-2021 offers labelled encrypted traffic, while many of the other well-known datasets do not. Capturing entire payload might be valuable in the future.
Anonymity	To maintain privacy, real traffic must anonymize specific packets, whereas synthetic traffic should give full packet capture.
Ground-truth	When compared to synthetic traffic, the datasets must deliver realistic traffic from a real production network and ensure no unlabeled attack within the ground-truth.
Up to date	By continuing the network traffic capturing procedure, both packet traces from flow data and pcap should be always accessible. The data may change over time, so repeating the procedures ensures that the dataset always has the most up-to-date information.
Labeled dataset	It is essential for precise and reliable analysis to

	appropriately identify data as malicious or benign. The labelling procedure is done manually and is defined by the source IP address, source port, destination IP address, destination port, and protocol of the flow.
Encryption Information	It must be specified how to create benign or harmful traffic. Effort is focused on application layer attacks such as brute force and probing that leverage HTTPS with TLS version 1.2 to deliver the attacks.

The author also provided a comparison between 12 other IDS datasets in terms of the contents and traits of the datasets which are displayed in **Table 4.2**.

Table 4.2: HIKARI-21 comparison with other IDS datasets.

Dataset	Comp. Capture	Payload	Anonymity	Ground-Truth	Up to Date Traffic	Labeled	Encryption	Practical to Generate
KDD99	Yes	Yes	No	Yes	No	Yes	No	No
MAWILab	Yes	No	Yes	No	Yes	Yes	Yes	Yes
CAIDA	Yes	No	Yes	No	Yes	No	No	No
SimpleWeb	Yes	No	No	No	No	No	No	Yes
NSL-KDD	Yes	Yes	No	Yes	No	Yes	No	No
IMPACT	Yes	No	Yes [1]	No	Yes	No	No	No
UMass	Yes	Yes	-	No	No	No	No	No
Kyoto	Yes	Yes	No	No	No	Yes	No	Yes
IRSC	Yes	Yes	No	Yes	No	Yes	No	No
UNSW-NB15	Yes	Yes	No	Yes	No	Yes	No	No
UGR’16	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes

Dataset	Comp. Capture	Payload	Anonymity	Ground-Truth	Up to Date Traffic	Labeled	Encryption	Practical to Generate
CICIDS-2017	Yes	Yes	Yes	No	No	Yes	Yes [2]	No

The author concluded that for an IDS dataset, there should be encrypted samples of benign and attack traffic. Payload from packet traces should be included. Moreover, the diversity of attack in terms of applications and protocols should be increased. The author also found that many of the datasets lack anonymized data. This may be caused by the datasets either getting developed in a controlled environment or getting permission when conducting the dataset generation activity. In addition, most of the datasets do not provide ground-truth data and background traffic. This could mean that analysis will be limited to their model. Due to the network environment being everchanging over time, it is important to have a methodology on developing a new dataset. Practical implementation when creating new datasets should be considered [5].

4.2 Dataset Features and Classes

The dataset contains data on application layer attacks that utilizes HTTPS and data of benign traffics. There are two categories of benign traffic (Benign or Background. While attack traffic has four categories which are Bruteforce, Bruteforce-XML, Probing, and XMRIGCC CryptoMiner. The traffic_category distribution in HIKARI-21 is shown in **Figure 4.1**.

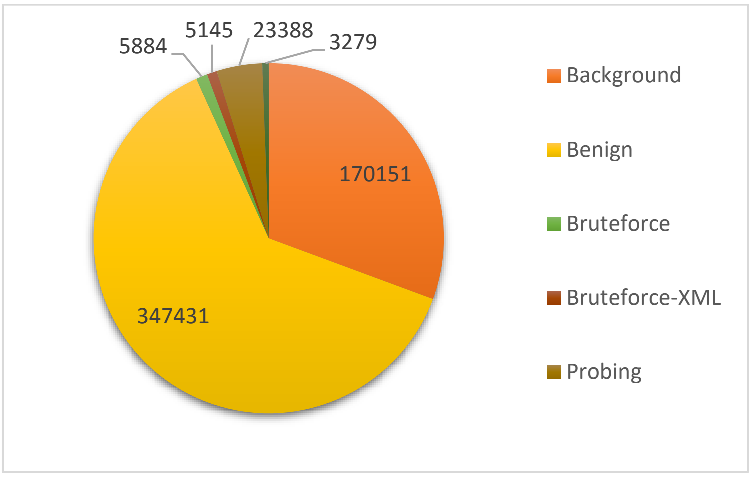


Figure 4.1: HIKARI-21 target class (traffic_category) distribution.

i. Benign

The benign traffic was generated with a profile similar to human behavior in mind. To create it, Selenium was utilized to run two headless browsers (Google Chrome and Mozilla Firefox). The two browsers mimic human behavior by clicking random links on multiple websites, registering as a user, signing in, uploading an article to the victim's server, and signing out. To avoid being identified as a bot or web spider, multiple configurations are used, such as user-agent and random delay, for each action sequence. Addresses are derived from Alexa’s top 1 million visitors list [https://www.alexa.com/]. Every benign traffic payload are not anonymized and are HTTPS.

ii. Background

Without a filter or firewall in the victim's network, all data for the background traffic are captured. Consequently, there is a chance that the background traffic contains malicious traffic or attacks. Several pieces of information, such as the IP address and payload, were anonymized to protect privacy without degrading the analysis's results.

iii. **Attack traffic**

Synthetically generated attack traffics are created by targeting user login page of the target websites (Bruteforce) and by scanning the websites for vulnerability (Probing). HTTPS is the protocol used for both attacks.

a) **Bruteforce**

In a bruteforce attack, an attacker will attempt to use many combinations of passwords through the use of dictionary list. To achieve this, a bruteforce script is developed.

b) **Bruteforce-XML**

Similar to bruteforce attack but using a XMLRPC attack vector instead of the browser attack vector.

c) **Probing**

Probes for vulnerability. The web applications get scanned for vulnerability using probing scripts, droopescan and joomscan.

d) **XMRIGCC CryptoMiner**

A malicious cryptomining traffic discovered to be a part of the background traffic during traffic validation.

HIKARI-21 consists of 555278 entries and 86 features. The entries are each labelled with the *traffic_category* and *Label* features to show the type of traffic and whether the flow is benign or malicious. The features in the dataset are listed in **Table 4.3**.

Table 4.3: HIKARI-21 features.

No	Feature	No	Feature	No	Feature
1	uid	30	flow_ECE_flag_count	59	flow_iat.avg
2	originh	31	fwd_pkts_payload.min	60	flow_iat.std
3	originp	32	fwd_pkts_payload.max	61	payload_bytes_per_second
4	responh	33	fwd_pkts_payload.tot	62	fwd_subflow_pkts
5	responp	34	fwd_pkts_payload.avg	63	bwd_subflow_pkts
6	flow_duration	35	fwd_pkts_payload.std	64	fwd_subflow_bytes
7	fwd_pkts_tot	36	bwd_pkts_payload.min	65	bwd_subflow_bytes
8	bwd_pkts_tot	37	bwd_pkts_payload.max	66	fwd_bulk_bytes
9	fwd_data_pkts_tot	38	bwd_pkts_payload.tot	67	bwd_bulk_bytes
10	bwd_data_pkts_tot	39	bwd_pkts_payload.avg	68	fwd_bulk_packets
11	fwd_pkts_per_sec	40	bwd_pkts_payload.std	69	bwd_bulk_packets
12	bwd_pkts_per_sec	41	flow_pkts_payload.min	70	fwd_bulk_rate
13	flow_pkts_per_sec	42	flow_pkts_payload.max	71	bwd_bulk_rate
14	down_up_ratio	43	flow_pkts_payload.tot	72	active.min
15	fwd_header_size_tot	44	flow_pkts_payload.avg	73	active.max
16	fwd_header_size_min	45	flow_pkts_payload.std	74	active.tot
17	fwd_header_size_max	46	fwd_iat.min	75	active.avg
18	bwd_header_size_tot	47	fwd_iat.max	76	active.std
19	bwd_header_size_min	48	fwd_iat.tot	77	idle.min
20	bwd_header_size_max	49	fwd_iat.avg	78	idle.max
21	flow_FIN_flag_count	50	fwd_iat.std	79	idle.tot

No	Feature	No	Feature	No	Feature
22	flow_SYN_flag_count	51	bwd_iat.min	80	idle.avg
23	flow_RST_flag_count	52	bwd_iat.max	81	idle.std
24	fwd_PSH_flag_count	53	bwd_iat.tot	82	fwd_init_window_size
25	bwd_PSH_flag_count	54	bwd_iat.avg	83	bwd_init_window_size
26	flow_ACK_flag_count	55	bwd_iat.std	84	fwd_last_window_size
27	fwd_URG_flag_count	56	flow_iat.min	85	**traffic_category**
28	bwd_URG_flag_count	57	flow_iat.max	86	**Label**
29	flow_CWR_flag_count	58	flow_iat.tot		

CHAPTER 5

IMPLEMENTATION

5.1 Data preprocessing

Dataset preparation starts by loading the dataset into a pandas Dataframe in Google Colab. The Dataframe allows us to prepare the data for the model by removing any meaningless columns. Prior to that, the required libraries for the project have been imported as well. The objective for this part is to find any missing value and correct any wrong data type for each feature. After checking, the datatypes are correct and there are no missing values throughout the dataset.

5.2 Imbalanced Dataset

HIKARI-21 is an imbalanced dataset, where the majority classes (Benign and Background) have significantly more instances compared to the minority classes of attack traffics. When training a classification model with an imbalanced dataset, the model's performance can be influenced by the distribution of classes in the dataset. This will cause for bias in the model where the majority classes are prioritized while the minority classes are overlooked [2]. Hence, it is expected that the multi-class classification decision tree model might produce unsatisfactory results.

5.3 Train Test split

Train test split is important as we need to have a separate set of data for training and testing to see how the model performs on new data after training. Essentially, the dataset is split according to a determined ratio, where one subset of the dataset will be used for training while the other for testing. In this project, the *train_test_split()* method from *sklearn* will be use to split the dataset. A test size of 30%, and train size of 70% are set for all models.

5.4 Multi-class Classification Results

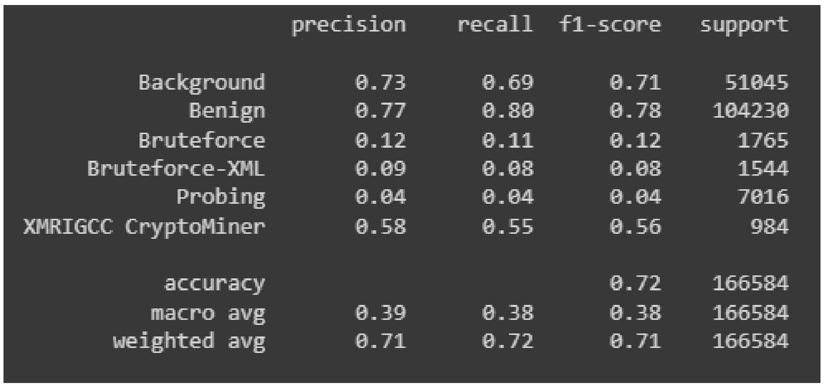
Multi-class classification is conducted by training the model to predict all classes available in the dataset. Note that the class distribution in training set and test set is similarly imbalanced. To evaluate the model's performance, a classification report and confusion matrix is produced, as shown in **Table 5.1, Figure 5.1,** and **Figure 5.2**.

Table 5.1: Score results

Model	Features	Accuracy	Precision	Recall	F1 score	Support
dt_default	79	0.70	0.41	0.45	0.43	166584

5.4.1 dt_default: Multi-class Classification with imbalanced distribution dataset.

The scores of dt_default shown in **Figure 5.1** shows the scores of the model such as accuracy score, precision score, recall score, and F1 score. The results produced are unsatisfactory as judging from the confusion matrix shown in Figure 5.9, the model failed to predict many instances of classes like Bruteforce, Bruteforce-XML, Probing, XMRIGCC Cryptominer correctly.

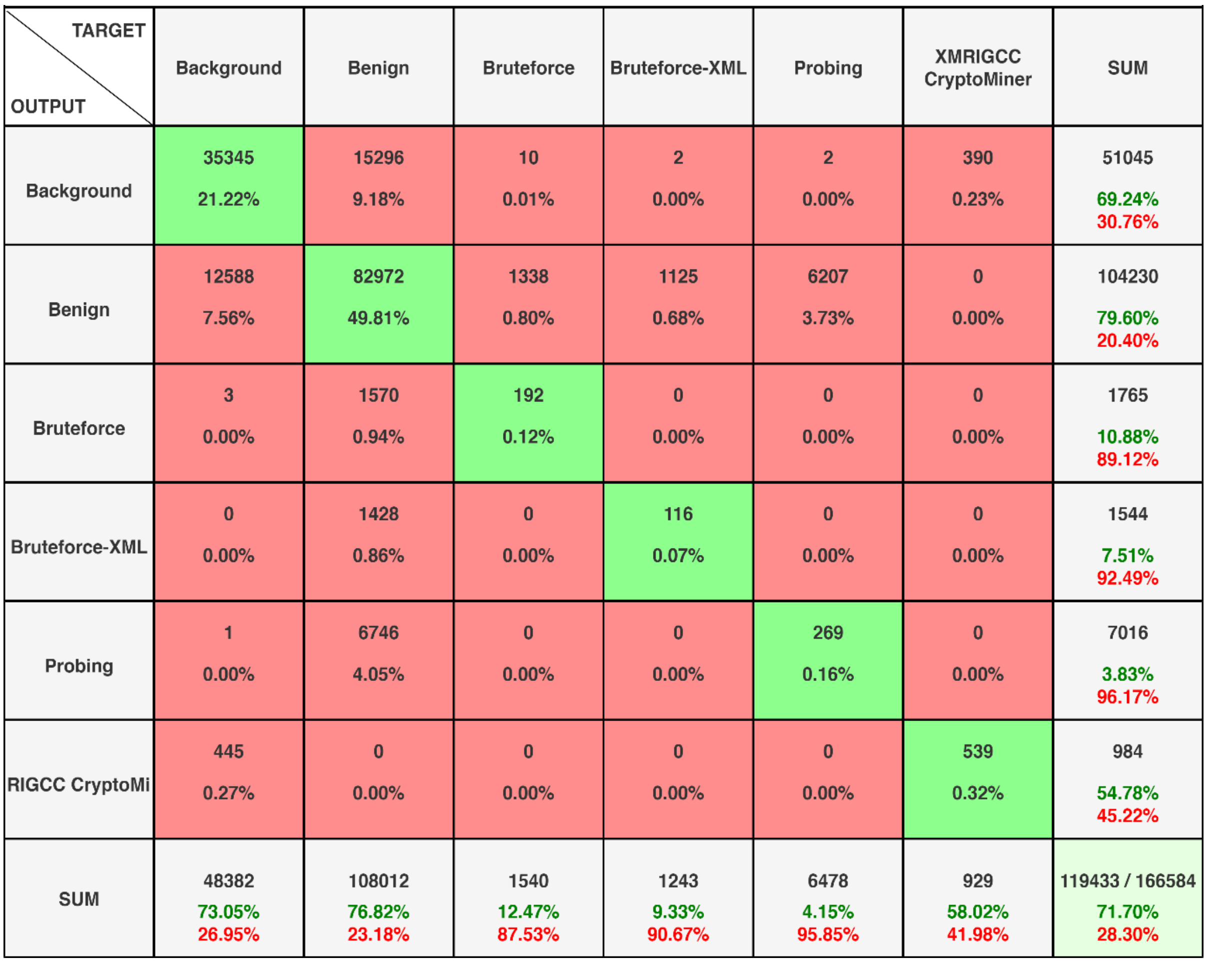


```
                     precision    recall  f1-score   support

          Background       0.73      0.69      0.71     51045
              Benign       0.77      0.80      0.78    104230
          Bruteforce       0.12      0.11      0.12      1765
      Bruteforce-XML       0.09      0.08      0.08      1544
             Probing       0.04      0.04      0.04      7016
 XMRIGCC CryptoMiner       0.58      0.55      0.56       984

            accuracy                           0.72    166584
           macro avg       0.39      0.38      0.38    166584
        weighted avg       0.71      0.72      0.71    166584
```

Figure 5.1: dt_default classes score results.



TARGET / OUTPUT	Background	Benign	Bruteforce	Bruteforce-XML	Probing	XMRIGCC CryptoMiner	SUM
Background	35345 21.22%	15296 9.18%	10 0.01%	2 0.00%	2 0.00%	390 0.23%	51045 69.24% 30.76%
Benign	12588 7.56%	82972 49.81%	1338 0.80%	1125 0.68%	6207 3.73%	0 0.00%	104230 79.60% 20.40%
Bruteforce	3 0.00%	1570 0.94%	192 0.12%	0 0.00%	0 0.00%	0 0.00%	1765 10.88% 89.12%
Bruteforce-XML	0 0.00%	1428 0.86%	0 0.00%	116 0.07%	0 0.00%	0 0.00%	1544 7.51% 92.49%
Probing	1 0.00%	6746 4.05%	0 0.00%	0 0.00%	269 0.16%	0 0.00%	7016 3.83% 96.17%
RIGCC CryptoMi	445 0.27%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	539 0.32%	984 54.78% 45.22%
SUM	48382 73.05% 26.95%	108012 76.82% 23.18%	1540 12.47% 87.53%	1243 9.33% 90.67%	6478 4.15% 95.85%	929 58.02% 41.98%	119433 / 166584 71.70% 28.30%

Figure 5.2: dt_default confusion matrix.

5.4.2 Comparison with other algorithms.

Table 5.2: Score results comparison.

Algorithm	Accuracy	Precision	Recall	F1
KNN	0.98	0.86	0.90	0.88
MLP	0.99	0.99	0.99	0.99
SVM	0.99	0.99	0.98	0.99
RF	0.99	0.99	0.99	0.99
dt_default	0.72	0.38	0.38	0.38

Table 5.2 displays the evaluation scores comparison between the three decision tree models and other algorithms tested by the creators of HIKARI-21 [5] dataset which are K-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), Support vector machine (SVM), and Random Forest (RF). The decision tree model developed were unable to achieve evaluation scores that is as high as other algorithms.

5.5 Binary Classification

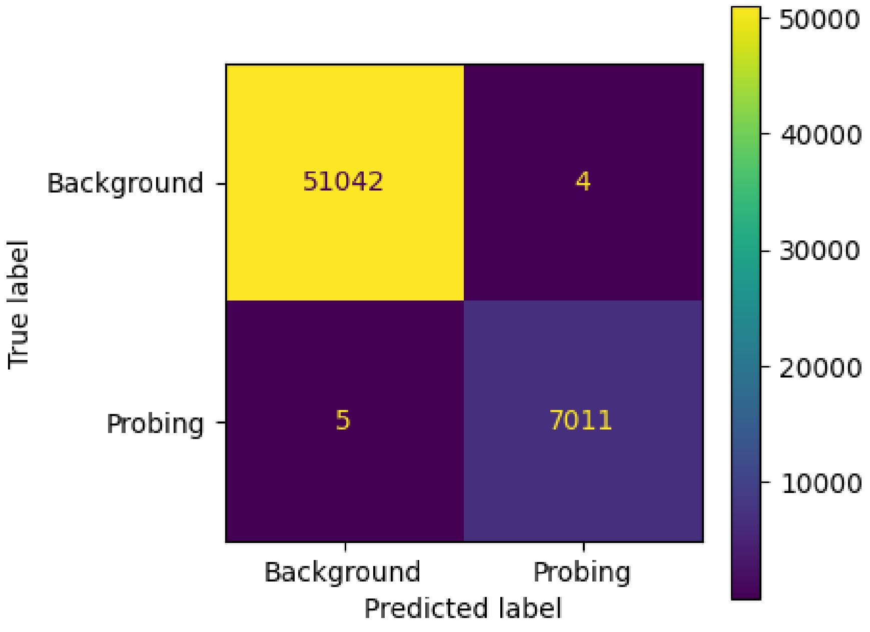
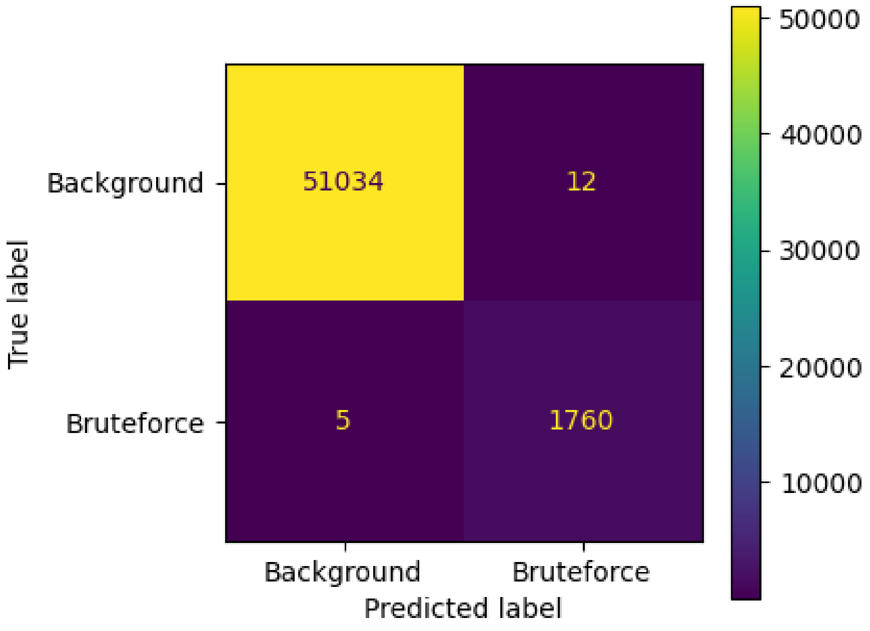
Binary classification is conducted by creating a model that only predicts between two classes (non-malicious and malicious) to see the performance of decision tree using HIKARI-21 for binary classification. Background is reduced from 170151 to 23821 only against XMRIGCC CryptoMiner for a more balanced distribution. The results of the binary classification decision tree models are shown in **Table 5.3** while confusion matrix shown in **Table 5.4**.

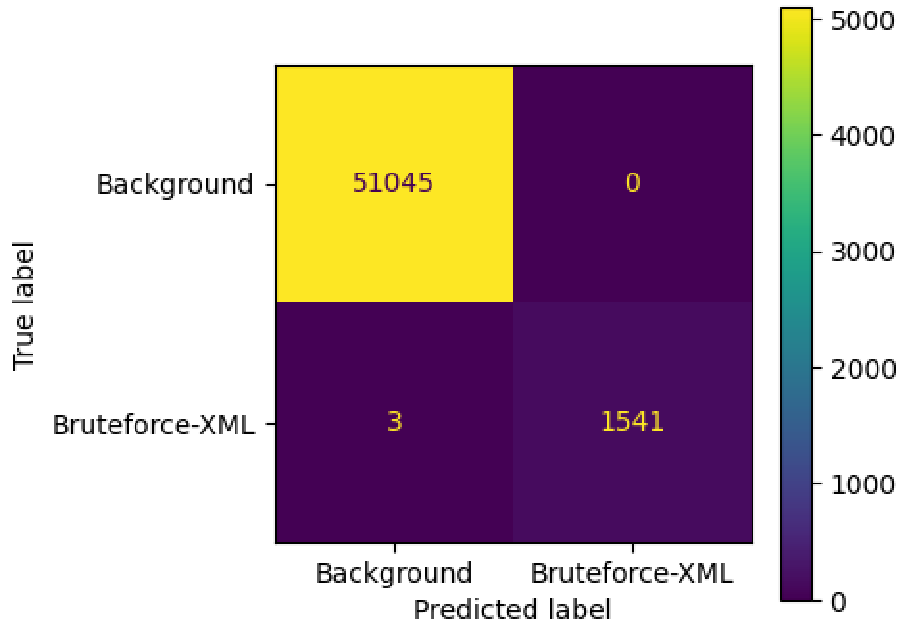
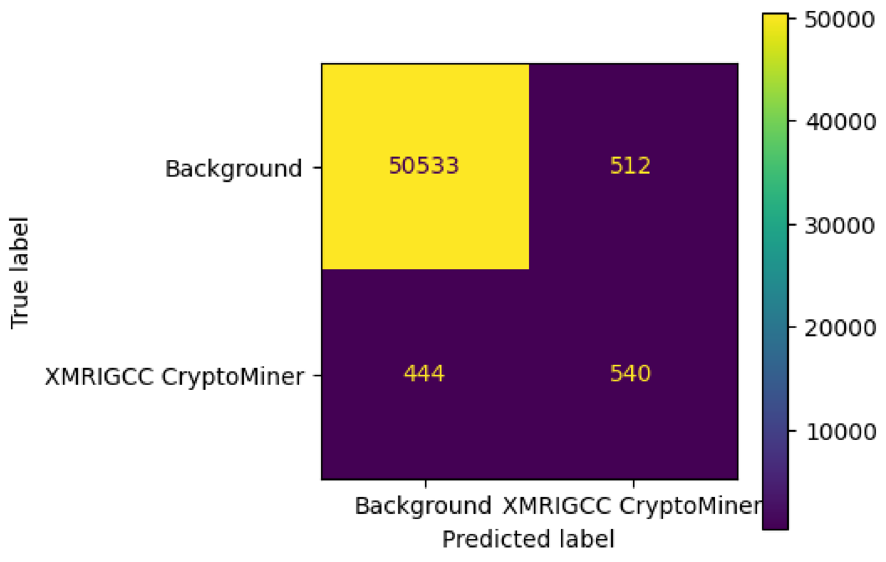
Table 5.3: Score results comparison for binary classification of Background against Malicious traffic.

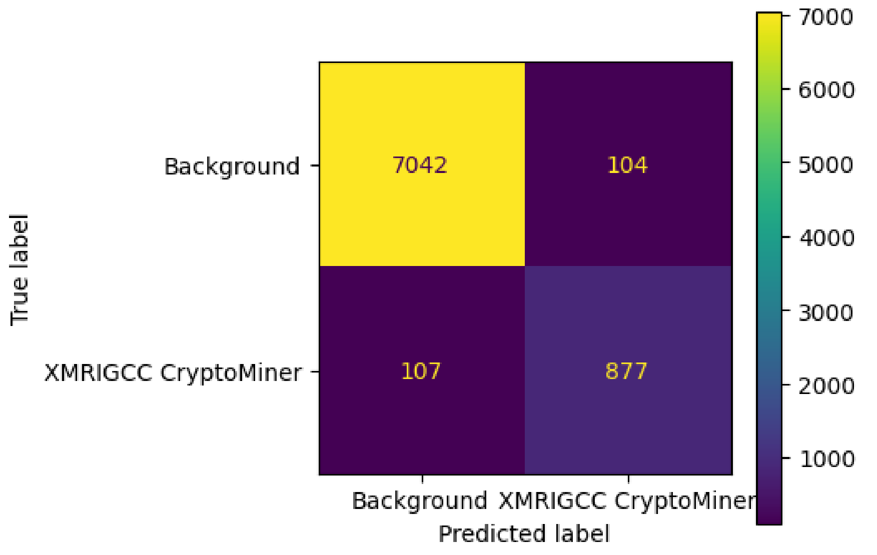
Model	Accuracy	Precision	Recall	F1
Background (170151) and Probing (23388)	1.00	1.00	1.00	1.00
Background (170151) and Bruteforce (5884)	1.00	1.00	1.00	1.00
Background (170151) and Bruteforce-XML (5145)	1.00	1.00	1.00	1.00
Background (170151) and XMRIGCC CryptoMiner (3279)	0.98	0.75	0.77	0.76

Background (23821) and XMRIGCC CryptoMiner (3279)	0.97	0.94	0.94	0.94

Table 5.4: Score results comparison for binary classification of Background against Malicious traffic.

Model	Confusion Matrix
Background (170151) and Probing (23388)	
Background (170151) and Bruteforce (5884)	

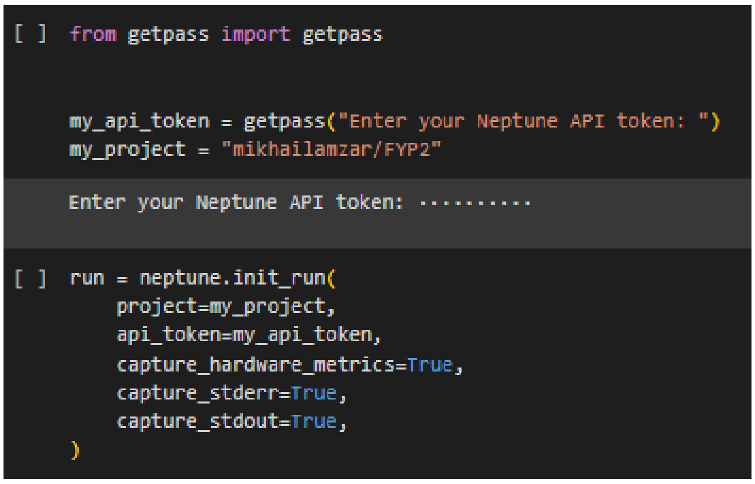
Background (170151) and Bruteforce-XML (5145)	
Background (170151) and XMRIGCC CryptoMiner (3279)	

Background (23821) and XMRIGCC CryptoMiner (3279)	

The binary classification decision tree models scores shown in **Table 5.3** are significantly better than multi-class classification. Confusion matrix in **Table 5.4** shows binary classification decision tree models prediction according to classes.

5.6 Dashboard in Neptune.ai

Neptune.ai is integrated with the Google Colab runtime in the same python notebook environment. The code to establish connection from Colab to Neptune.ai is shown in **Figure 5.3.**



```
[ ]  from getpass import getpass


     my_api_token = getpass("Enter your Neptune API token: ")
     my_project = "mikhailamzar/FYP2"

     Enter your Neptune API token: ··········

[ ]  run = neptune.init_run(
         project=my_project,
         api_token=my_api_token,
         capture_hardware_metrics=True,
         capture_stderr=True,
         capture_stdout=True,
     )
```

Figure 5.3: Establishing connection to Neptune.ai from Google Colab

After connection is established, models in Google Colab can be developed until finished. The evaluation scores are extracted from the models and sent to Neptune.ai with the following code shown in **Table 5.5.**

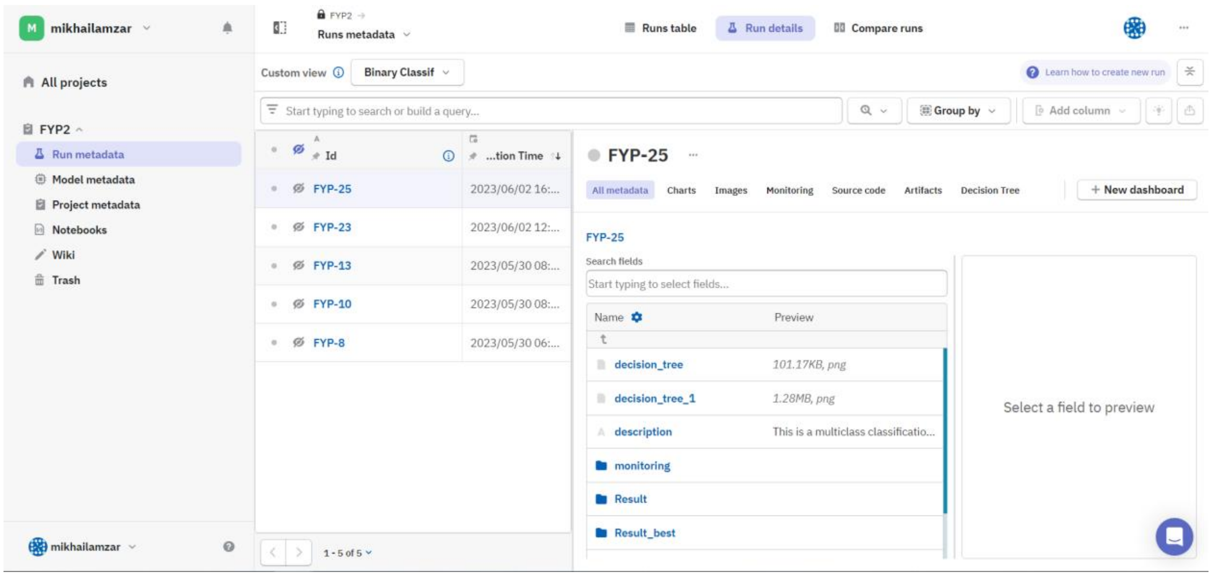
Table 5.5: Code to send model evaluation score to Neptune.ai

Description	Code

Assigning new score variable with values extracted from score attributes from the decision tree model.	from sklearn.metrics import accuracy_score
	from sklearn.metrics import f1_score
	from sklearn.metrics import precision_score
	from sklearn.metrics import recall_score
	
	acc = accuracy_score(y_test,y_pred)
	f1 = f1_score(y_test,y_pred, average='macro')
	pre = precision_score(y_test,y_pred, average='macro')
	rec = recall_score(y_test,y_pred, average='macro')

Sending the new score variables to the runtime logging on Neptune.ai.	run["train_dataset"].track_files ("/content/drive/MyDrive/ALLFLOWMETER_HIKARI2021.csv") run["description"] = "Write desc here" # score run["Result/score"] = report run["Result/accuracy"] = acc run["Result/f1-score"] = f1 run["Result/precision"] = pre run["Result/recall"] = rec run["Result/classif_report"].upload(CR) #run["decision_tree"].upload('dt_tree.png') run["Result/confusion_matrix"] = fig run["decision_tree"].upload(tree_fig)

The dashboard page is shown in **Figure 5.4.** The models are monitored on a runtime basis, shown in 'Run Metadata' tab. **Figure 5.5** shows the evaluations scores and other details stored.

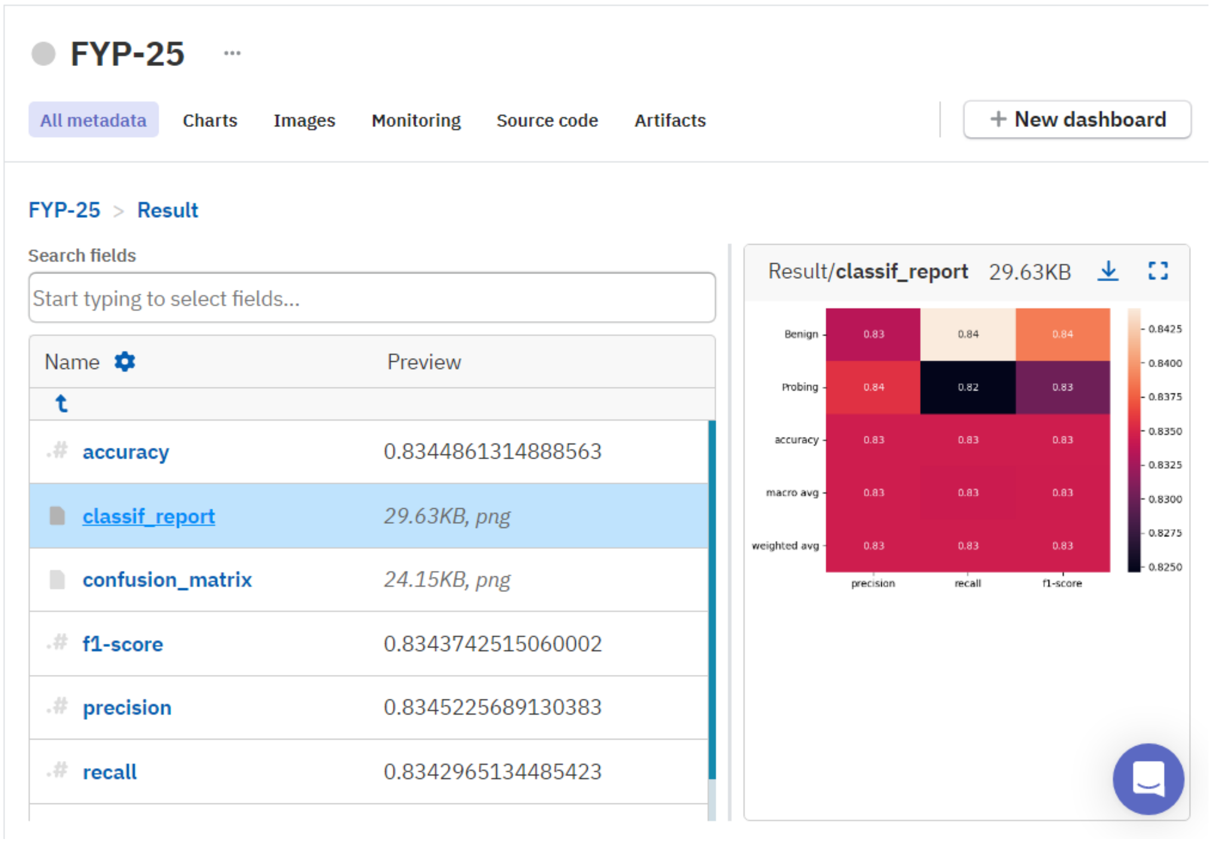


Id	...tion Time
FYP-25	2023/06/02 16:...
FYP-23	2023/06/02 12:...
FYP-13	2023/05/30 08:...
FYP-10	2023/05/30 08:...
FYP-8	2023/05/30 06:...

FYP-25

Name	Preview
decision_tree	101.17KB, png
decision_tree_1	1.28MB, png
description	This is a multiclass classificatio...
monitoring	
Result	
Result_best	

Figure 5.4: Neptune.ai dashboard page



Name	Preview
accuracy	0.8344861314888563
classif_report	29.63KB, png
confusion_matrix	24.15KB, png
f1-score	0.8343742515060002
precision	0.8345225689130383
recall	0.8342965134485423

	precision	recall	f1-score
Benign	0.83	0.84	0.84
Probing	0.84	0.82	0.83
accuracy	0.83	0.83	0.83
macro avg	0.83	0.83	0.83
weighted avg	0.83	0.83	0.83

Figure 5.5: Neptune.ai evaluation score log

CHAPTER 6

CONCLUSION

In this paper, we performed several techniques to develop a network intrusion detection model using decision tree algorithm based on HIKARI-21 real and encrypted synthetic attack traffic dataset. The project objectives which are to design a network intrusion detection model that applies Decision Tree algorithm and to produce a machine learning model and dashboard for network intrusion detection model are satisfied. Firstly, A multi-class classification decision tree model was developed which used the imbalanced data for training and testing. The result of the multiclass model is unsatisfactory because of the poor evaluation scores and confusion matrix that clearly shows the model's poor performance to correctly predict classes in the dataset. Therefore, HIKARI-21 with the default class distribution could not be used to develop a good multi-class decision tree model. To improve this in future works, the dataset class instances could be modified by changing the numbers of classes to be more evenly distributed, leading to a more balanced distribution of classes. One of the most common ways to achieve this is through resampling techniques. Secondly, binary classification decision tree models that were developed in this project yielded better results compared to multi-classification. Surprisingly, even with the imbalanced distribution between two classes, such as Background (170151) and Probing (23388), Background (170151) and Bruteforce (5884), and

Background (23821) and XMRIGCC CryptoMiner (3279), the models produced significantly better evaluation scores compared to the multi-class model. This result shows that HIKARI-21 is a viable dataset for a network intrusion detection model using binary classification decision tree algorithm. The evaluation scores in this project are also successfully logged and stored using Neptune.ai machine learning platform. The integration of Neptune.ai with Google Colab in this project allows us to conveniently store important results and data from the decision tree models to ease the process of analytical observations and comparison. This implementation of Neptune.ai could also be utilized for other machine learning project using other types of algorithms to ease analysis and observation activity.

REFERENCES

1. J. K. Chahal, V. Gandhi, P. Kaushal, K. R. Ramkumar, A. Kaur and S. Mittal, "KAS-IDS: A Machine Learning based Intrusion Detection System," 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 2021, pp. 90-95, doi: 10.1109/ISPCC53510.2021.9609402.
2. N. Cahyana, S. Khomsah and A. S. Aribowo, "Improving Imbalanced Dataset Classification Using Oversampling and Gradient Boosting," 2019 5th International Conference on Science in Information Technology (ICSITech), Yogyakarta, Indonesia, 2019, pp. 217-222, doi: 10.1109/ICSITech46713.2019.8987499.
3. Kunal and M. Dua, "Machine Learning Approach to IDS: A Comprehensive Review," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 117-121, doi: 10.1109/ICECA.2019.8822120.
4. C. Yu, F. Li, G. Li and N. Yang, "Multi-classes Imbalanced Dataset Classification Based on Sample Information," 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, New York, NY, USA, 2015, pp. 1768-1773, doi: 10.1109/HPCC-CSS-ICESS.2015.327.
5. Ferriyan A, Thamrin AH, Takeda K, Murai J. Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic. Applied Sciences. 2021; 11(17):7868. https://doi.org/10.3390/app11177868
6. S. P. Chytas, L. Maglaras, A. Derhab and G. Stamoulis, "Assessment of Machine Learning Techniques for Building an Efficient IDS," 2020 First International Conference of Smart Systems and Emerging Technologies

(SMARTTECH), Riyadh, Saudi Arabia, 2020, pp. 165-170, doi: 10.1109/SMART-TECH49988.2020.00048.

7. Accuracy, Precision, Recall & F1-Score. 2023. Internet: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example
8. Precision and Recall | Essential Metrics for Data Analysis. 2020. Internet: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning
9. Learning from Imbalanced Classes. 2016. Internet: https://www.svds.com/learning-imbalanced-classes/
10. The Basics of Classifier Evaluation: Part 2. 2015. Internet: http://www.svds.com/classifiers2/
11. The Basics of Classifier Evaluation: Part 1. 2015. Internet: https://www.svds.com/the-basics-of-classifier-evaluation-part-1/
12. Oversampling and Undersampling: A technique for Imbalanced Classification. 2020. Internet: https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf
13. Evaluation Metrics for Classification Problems with Implementation in Python. 2021. Internet: https://medium.com/analytics-vidhya/evaluation-metrics-for-classification-problems-with-implementation-in-python-a20193b4f2c3
14. Blagus, R., Lusa, L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 14, 106 (2013). https://doi.org/10.1186/1471-2105-14-106
15. How to Calculate Feature Importance With Python. 2020. Internet: https://machinelearningmastery.com/calculate-feature-importance-with-python/
16. Information Gain Computation in Python. Internet: https://www.featureranking.com/tutorials/machine-learning-tutorials/information-gain-computation/
17. U. Dixit, S. Bhatia and P. Bhatia, "Comparison of Different Machine Learning Algorithms Based on Intrusion Detection System," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing

(COM-IT-CON), Faridabad, India, 2022, pp. 667-672, doi: 10.1109/COM-IT-CON54601.2022.9850515.

18. Metrics to Evaluate your Classification Model to take the right decisions. 2021. Internet: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/
19. "Decision Trees." sklearn documentation. Internet: https://scikit-learn.org/stable/modules/tree.html
20. U. Dixit, S. Bhatia and P. Bhatia, "Comparison of Different Machine Learning Algorithms Based on Intrusion Detection System," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), Faridabad, India, 2022, pp. 667-672, doi: 10.1109/COM-IT-CON54601.2022.9850515.
21. Ali, Jehad & Khan, Rehanullah & Ahmad, Nasir & Maqsood, Imran. (2012). Random Forests and Decision Trees. International Journal of Computer Science Issues (IJCSI). 9.
22. What is a Decision Tree. Internet: https://www.ibm.com/topics/decision-trees
23. Decision Tree on Imbalanced Dataset. 2019. Internet: https://medium.com/@160shelf/decision-tree-on-imbalanced-dataset-f1575414a6c2
24. Oversampling and Undersampling: A technique for imbalanced classification. 2020. Internet: https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf
25. SMOTE for Imbalanced Classification with Python. 2021. Internet: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
26. Scikit-learn: Decision Trees. Internet: https://scikit-learn.org/stable/modules/tree.html#mathematical-formulation
27. "Agile Methodologies: A Comprehensive Guide." Atlassian. Internet: https://www.atlassian.com/agile
28. "The Agile Manifesto." Internet: https://agilemanifesto.org
29. Machine Learning Mastery. Internet: https://machinelearningmastery.com

30. “Applications of Machine Learning” Medium. Internet: https://medium.com/swlh/applications-of-machine-learning-9f77e6eb7acc

31. Yasir Hamid1, M. Sugumaran and V. R. Balasaraswathi . 2016. IDS Using Machine Learning – Current State of Art and Future Directions. *British Journal of Applied Science & Technology 15(3): 1-22, Article no.BJAST.23668 ISSN: 2231-0843, NLM ID: 101664541*

32. "Decision Tree." GeeksforGeeks. Internet: https://www.geeksforgeeks.org/decision-tree/

33. "Decision Trees." sklearn documentation. Internet: https://scikit-learn.org/stable/modules/tree.html

34. Google Colab “Getting Started”. Internet: https://colab.research.google.com

35. Grafana “Everything You Should Know About It”. Internet: https://scaleyourapp.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/

36. What Is Splunk? A Beginners Guide To Understanding Splunk. 2022. Internet: https://www.edureka.co/blog/what-is-splunk/

37. FireEye Network Threat Prevention Platform. Internet: https://www.fireeye.com/content/dam/fireeye-www/products/pdfs/pf/web/fireeye-network-threat-prevention-platform.pdf

38. [Anaconda Documentation. Internet: https://docs.anaconda.com/navigator/index.html

39. What is Weka. Internet: https://www.tutorialspoint.com/weka/what_is_weka.htm

40. What is .NET. Internet: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

41. M. Somvanshi, P. Chavan, S. Tambade and S. V. Shinde, "A review of machine learning techniques using decision tree and support vector machine," 2016 International Conference on Computing Communication Control and automation (ICCUBEA), 2016, pp. 1-7, doi: 10.1109/ICCUBEA.2016.7860040.

42. A. Navada, A. N. Ansari, S. Patil and B. A. Sonkamble, "Overview of use of decision tree algorithms in machine learning," 2011 IEEE Control and System Graduate Research Colloquium, 2011, pp. 37-42, doi: 10.1109/ICSGRC.2011.5991826.

43. SDLC – Waterfall Model. Internet: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

44. Abid Ali Awan, 2022. The Machine Learning Life Cycle Explained. Internet: https://www.datacamp.com/blog/machine-learning-lifecycle-explained

45. Jafar Alzubi et al 2018 J. Phys.: Conf. Ser. 1142 012012

46. Ferriyan, A.; Thamrin, A.H.; Takeda, K.; Murai, J. Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic. Appl. Sci. 2021, 11, 7868. https://doi.org/10.3390/app11177868

47. Seldon, 2021, Decision Tree In Machine Learning. Internet: https://www.seldon.io/decision-trees-in-machine-learning \

48. Bhuvaneswari Gopalan, 2020. “Is Decision Tree a classification or regression model?” Internet: https://www.numpyninja.com/post/is-decision-tree-a-classification-or-regressionmodel