National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity

Charles Doherty
Student ID: x23287039

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Charles Doherty

Student ID: x23287039

Programme: MSc Cybersecurity **Year:** 2025

Module: Practicum Part 2 (Configuration Manual)

Lecturer: Ross Spelman

Submission Due Date: 15 September 2025

Project Title: Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning (Configuration Manual)

Word Count: 3,270 **Page Count:** 15 (excl. References)

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Charles Doherty

Date: 25 August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual
Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning

Charles Doherty
Student ID: x23287039

Table of Contents

1 Configuration Manual Overview

This configuration manual is intended to provide guidance for reproducing the work carried out during research activities for this MSc Research Project. The title of the accompanying research report is “Optimising Internet of Medical Things Intrusion Detection Performance using Machine Learning”.

The code is written in Python and uses the Miniconda3 (conda) package and environment manager for installation of various packages and tools used as part of the research. Installation of the required software environment and preparation of the dataset are presented in Sections 2 and 3 following this overview.

All code is run in a Jupyter Notebook by executing cells as required, with detailed instructions on the operation of the code provided in Sections 4-7 of this manual.

2 System Configuration

This section outlines the details of hardware and software used during the research activities. The environment can be re-created on another system by following the instructions included in this section, allowing for ease of reproducibility of results. Note that due to differences in processing power, RAM and system resource usage, code may take a varying amount of time to execute across systems, so the time taken for code execution will change proportionately according to the system it is executed on.

2.1 Hardware Specifications and Software Versions

Hardware/OS Specifications:

Component	Specification
Operating System	Windows 11 Pro (24H2)
CPU	13th Gen Intel(R) Core(TM) i7-1360P 2.20 GHz, 12 Cores
RAM	16GB

Application software:

Software	Version	Purpose
Miniconda3	py312_24.7.1-0	Python package and environment management
Mozilla Firefox	139.0.4	Web Browser used to run Jupyter Notebooks

Key Python Packages and Libraries installed in conda environment:

Package	Version	Purpose
python	3.13.2	Programming language chosen for this research project
jupyter	Multiple packages - see .yml file for versions	Web-based interactive environment for executing python code for machine learning
numpy	2.2.4	Library for working with large arrays in Python
pandas	2.2.3	Python data analysis library
scikit-learn	1.6.1	Provides libraries for machine learning functionality in Python. Full documentation for chosen algorithms can be found on their website[1].
seaborn	0.13.2	Statistical data visualisation package for python
matplotlib	3.10.0	Python visualisation library
tabulate	0.9.0	Provides output formatting for results data
bayesian-optimization	2.0.4	Python implementation of Bayesian Optimisation used for Hyperparameter Tuning

All python packages were installed via conda-forge, using the "conda install" command. Many additional packages were installed as dependencies on related to the table above. All packages and versions can be seen in the "x23287039_iomtml_env.yml" file included with the submission, which is an export of the conda environment used for all activities during this research.

2.2 Re-creating the software environment

The environment named "iomtml" used during research can be created on any Microsoft Windows system by following the instructions below.

1. Install Miniconda3.
2. Ensure a modern web browser is installed on the system. Jupyter Notebook documentation recommends using Firefox, Safari or Chrome to ensure compatibility.
3. Open an Anaconda PowerShell Prompt (miniconda3) on the system.
4. Import the "x23287039_iomtml_env.yml" file (provided in the submission) using the following command:

   conda env create --file **<local_filepath>**\x23287039_iomtml_env.yml

[1] Scikit-learn Documentation: https://scikit-learn.org/stable/index.html

2.3 Open Jupyter Notebook with Research Code

Once the environment has been created, the Jupyter Notebook submitted with code used for research named "x23287039_Thesis_Code.ipynb" can be opened by performing the following steps:

1. Switch to the newly created environment using the command:
   conda activate iomtml
2. Open Jupyter Notebook using the following command:
   jupyter notebook
3. A browser window will automatically open with the URL "http://localhost:8888/tree".
4. Browse to the "x23287039_Thesis_Code.ipynb" file included with the submission.
5. The Jupyter Notebook will open in a separate browser tab, ready for code to be executed.

An additional file named "x23287039_Thesis_Code_with_Sample_Outputs.ipynb" is also included with the submission for demonstration purposes. This file contains identical code to the "x23287039_Thesis_Code.ipynb" file, but includes outputs generated from previous code executions included for demonstration purposes. In the "x23287039_Thesis_Code.ipynb" file, all outputs have been cleared.

2.4 Notes on Jupyter Notebook Code

Print statements in Python were used throughout the development of the code in order to output to the screen to ensure that the code was being executed as desired. The most useful of these were left in the final code, as they provided feedback on code execution during the gathering of results data. In some cases, print statements were commented out using the "#" character at the beginning of the statement, but these may be uncommented if the user wishes to view additional information outputted to the screen during code execution. Other useful lines of code that were used intermittently, such as different hyperparameter settings for classification algorithms, can also be commented out (adding a "#" at the start of a line) and uncommented in (removing the "#" at the start of a line) as desired. Additional comments exist for informational purposes to describe a section of code.

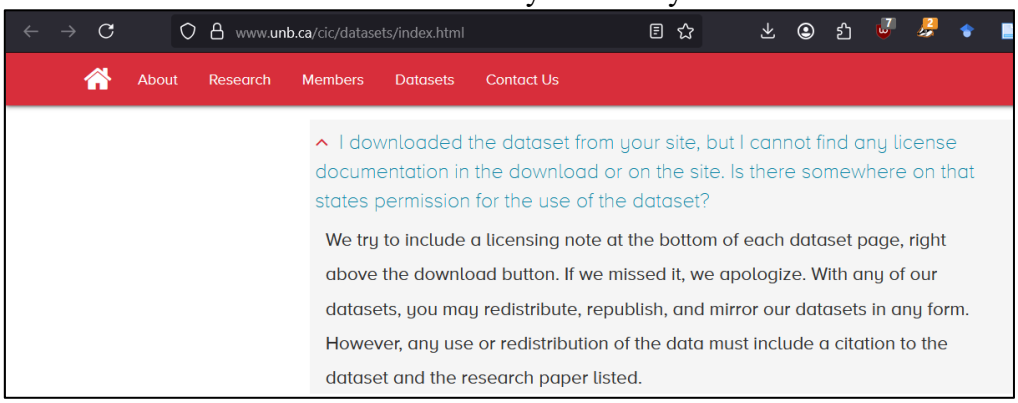
Cell groups in a Jupyter Notebook can be hidden or expanded as desired at different levels of the structure by using a marker to the left of the text headers. Code cells can be executed by selecting the cell and pressing Shift+Enter, or pressing the "Play" button as shown in the screenshot below. Prior to running code, ensure that "Python 3 (ipykernel)" is selected as the kernel, as shown on the right side of the screenshot. Further information on the use of Jupyter Notebooks can be found in their online documentation[2].

Many functions used throughout the code have a parameter called "random_state". This is used to assist with reproducibility when performing certain actions. For all instances where the parameter is used, the setting "random_state=7" has been applied.

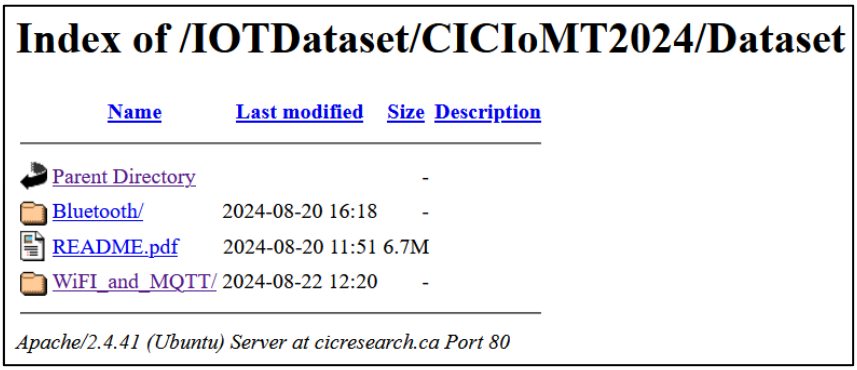
[2] Jupyter Notebook Documentation: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

3 Source Dataset

The CICIoMT2024 dataset (Dadkhah et al., 2024) used during this research is freely available for download on the Canadian Institute for Cybersecurity (CIC) website[3]. The full dataset (plus the additional file structure for creating the reduced dataset as detailed in section 3.1 and 3.2) could not be included with the code submission, as they have a total size of approximately 6.4GB. The dataset is freely available to use as per licensing terms outlined in the common questions section of the Canadian Institute for Cybersecurity website[4].



After selecting the download link you must submit a short form before being redirected to the site containing the dataset files.



All files located in the following folder must be downloaded to recreate the dataset locally:

/IOTDataset/CICIoMT2024/Dataset/WiFI_and_MQTT/attacks/CSV

Note: Due to a typo when initially creating the folders locally which was only discovered on creation of this manual, the **"attacks"** folder in the source must be called **"attack"** in the local folder path structure in order to run the code successfully. Additionally, the CSV folder was not created as there was no requirement to download the raw PCAP files, meaning there was no need to distinguish between both. These folder structure modifications have no impact on the data within the CSV files. The end result should be that all files within the source "/IOTDataset/CICIoMT2024/Dataset/WiFI_and_MQTT/attacks/CSV" train and test folders are downloaded to train and test folders in a local folder at the following location:

<local_filepath>\CICIoMT2024\Dataset\WiFI_and_MQTT\attack

[3] Download link for CICIoMT2024 Dataset: https://www.unb.ca/cic/datasets/iomt-dataset-2024.html

[4] Common Questions with licencing information for CIC Datasets: https://www.unb.ca/cic/datasets/index.html

After downloading the files locally, take note of the full file path (**<local_filepath>**) of the CICIoMT2024 folder locally for use in the code in section 4.2.1.

3.1 Reduced Dataset Creation

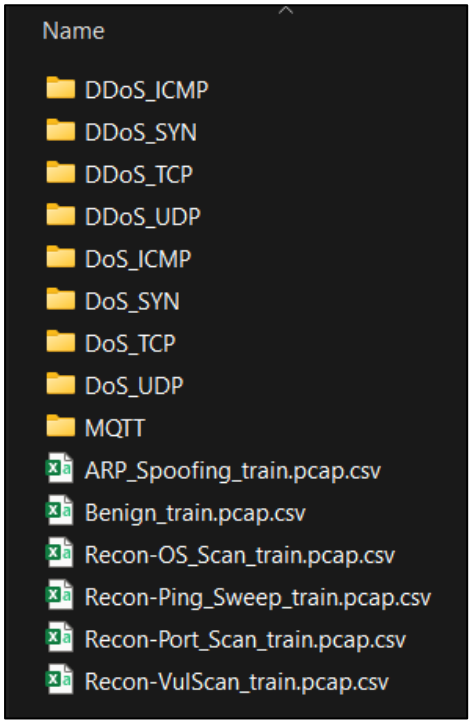
As the duration of processing times was proving problematic using the full dataset, a reduced set of data was created by manually copying the full dataset files into an additional folder structure for processing using python code. Two new folders were created, one for each of the train ("train_reduced") and test ("test_reduced") datasets.

3.1.1 Reduced Train Dataset

To recreate the training dataset, the structure in the screenshots below must be created in a new folder at the following location:

<local_filepath>\CICIoMT2024\Dataset\WiFI_and_MQTT\attack\train_reduced

```
C:.
    ARP_Spoofing_train.pcap.csv
    Benign_train.pcap.csv
    Recon-OS_Scan_train.pcap.csv
    Recon-Ping_Sweep_train.pcap.csv
    Recon-Port_Scan_train.pcap.csv
    Recon-VulScan_train.pcap.csv

├───DDoS_ICMP
        TCP_IP-DDoS-ICMP1_train.pcap.csv
        TCP_IP-DDoS-ICMP2_train.pcap.csv
        TCP_IP-DDoS-ICMP3_train.pcap.csv
        TCP_IP-DDoS-ICMP4_train.pcap.csv
        TCP_IP-DDoS-ICMP5_train.pcap.csv
        TCP_IP-DDoS-ICMP6_train.pcap.csv
        TCP_IP-DDoS-ICMP7_train.pcap.csv
        TCP_IP-DDoS-ICMP8_train.pcap.csv

├───DDoS_SYN
        TCP_IP-DDoS-SYN1_train.pcap.csv
        TCP_IP-DDoS-SYN2_train.pcap.csv
        TCP_IP-DDoS-SYN3_train.pcap.csv
        TCP_IP-DDoS-SYN4_train.pcap.csv

├───DDoS_TCP
        TCP_IP-DDoS-TCP1_train.pcap.csv
        TCP_IP-DDoS-TCP2_train.pcap.csv
        TCP_IP-DDoS-TCP3_train.pcap.csv
        TCP_IP-DDoS-TCP4_train.pcap.csv

├───DDoS_UDP
        TCP_IP-DDoS-UDP1_train.pcap.csv
        TCP_IP-DDoS-UDP2_train.pcap.csv
        TCP_IP-DDoS-UDP3_train.pcap.csv
        TCP_IP-DDoS-UDP4_train.pcap.csv
        TCP_IP-DDoS-UDP5_train.pcap.csv
        TCP_IP-DDoS-UDP6_train.pcap.csv
        TCP_IP-DDoS-UDP7_train.pcap.csv
        TCP_IP-DDoS-UDP8_train.pcap.csv

├───DoS_ICMP
        TCP_IP-DoS-ICMP1_train.pcap.csv
        TCP_IP-DoS-ICMP2_train.pcap.csv
        TCP_IP-DoS-ICMP3_train.pcap.csv
        TCP_IP-DoS-ICMP4_train.pcap.csv

├───DoS_SYN
        TCP_IP-DoS-SYN1_train.pcap.csv
        TCP_IP-DoS-SYN2_train.pcap.csv
        TCP_IP-DoS-SYN3_train.pcap.csv
        TCP_IP-DoS-SYN4_train.pcap.csv

├───DoS_TCP
        TCP_IP-DoS-TCP1_train.pcap.csv
        TCP_IP-DoS-TCP2_train.pcap.csv
        TCP_IP-DoS-TCP3_train.pcap.csv
        TCP_IP-DoS-TCP4_train.pcap.csv

├───DoS_UDP
        TCP_IP-DoS-UDP1_train.pcap.csv
        TCP_IP-DoS-UDP2_train.pcap.csv
        TCP_IP-DoS-UDP3_train.pcap.csv
        TCP_IP-DoS-UDP4_train.pcap.csv

└───MQTT
        MQTT-DDoS-Connect_Flood_train.pcap.csv
        MQTT-DDoS-Publish_Flood_train.pcap.csv
        MQTT-DoS-Connect_Flood_train.pcap.csv
        MQTT-DoS-Publish_Flood_train.pcap.csv
        MQTT-Malformed_Data_train.pcap.csv
```

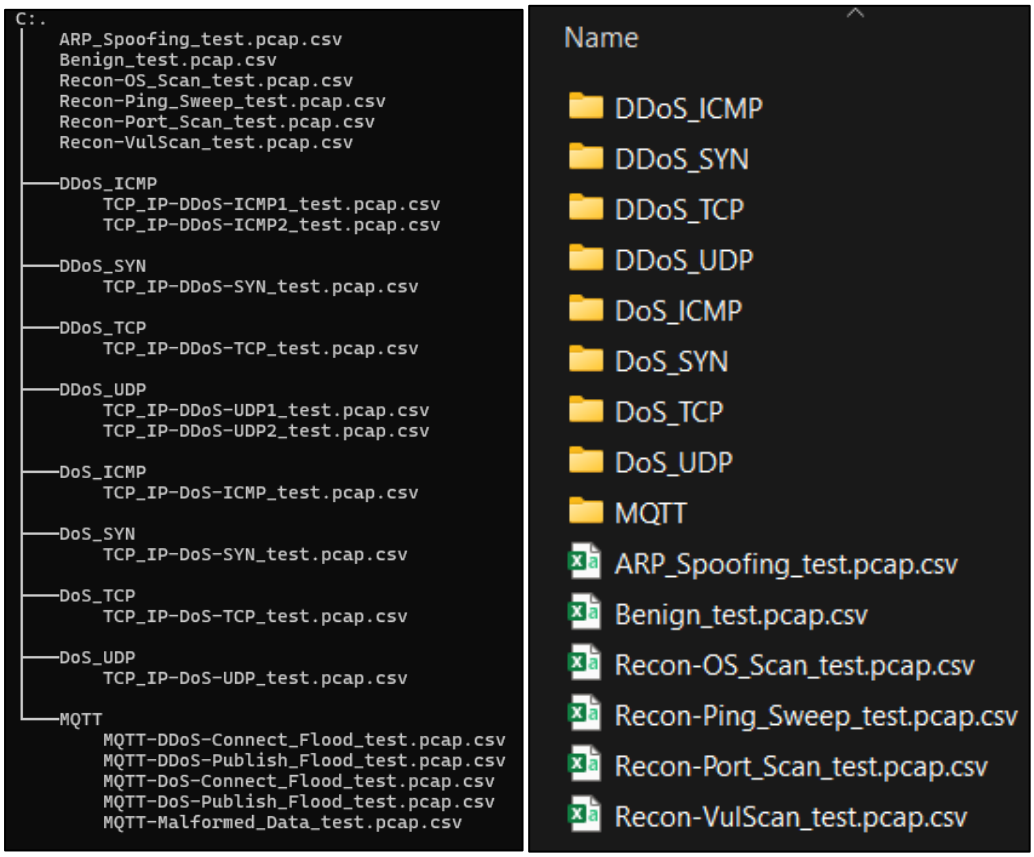


Source CSV files (names have not been altered) must be manually copied from the "**<local_filepath>**\CICIoMT2024\Dataset\WiFI_and_MQTT\attack\train" folder to the new "train_reduced" subfolder within the "attack" folder. The six files related to ARP, Benign, and Recon traffic sit directly within the top level of the "train_reduced" folder, with all other files in subfolders as shown in the tree structure above.

3.1.2 Reduced Test Dataset

For the reduced test dataset, a similar process is followed as performed for the train dataset. The file structure in the screenshots below must be created in a new folder at the following location:

<local_filepath>\CICIoMT2024\Dataset\WiFI_and_MQTT\attack\test_reduced



Source CSV files (with original names) must be manually copied from the "**<local_filepath>**\CICIoMT2024\Dataset\WiFI_and_MQTT\attack\test" folder to the new "test_reduced" subfolder within the "attack" folder. As before, the six files related to ARP, Benign, and Recon traffic sit directly within the top level of the "test_reduced" folder, with all other files in subfolders as shown in the tree structure above.

4 Importing Packages and Importing Dataset

4.1 Importing Packages

Jupyter Notebook Section Name: *"Package Imports"*

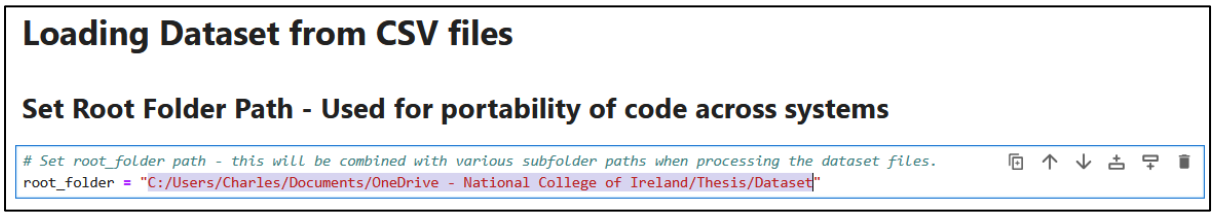
All packages required to run any of the code within the Jupyter Notebook are first imported using the "Package Imports" section of the code. This cell should be executed first after opening the notebook and prior to executing any other cells to avoid errors related to missing packages.

4.2 Dataset Import from Files

Jupyter Notebook Section Name: *"Loading Dataset from CSV files"*

4.2.1 Setting Folder Path

In order to recreate the file processing correctly on another system, the "root_folder" variable must be set correctly to match the location of the CICIoMT2024 Dataset files (download and reduced dataset creation is detailed in section 3) on the local system. Replace the highlighted text in the screenshot below with the applicable location:



4.2.2 Import Data into Train and Test Datasets

Depending on the desired use case, the various types of datasets may be imported into dataframes by executing the code in the relevant sections, which also labels the instances accordingly for Binary or Categorical classification scenarios. The code for importing the files was adapted from method 2 in an article[5] discussing options to read multiple CSV files into pandas, with additional information required for implementation gathered from a PYnative web page on Python glob[6]. Stratified sampling of files was performed based on examples outlined in a GeeksforGeeks article[7].

4.2.2.1 Binary Classification

Initial tests were run with the full original CICIoMT2024 Dataset. However, these tests were limited due to prohibitive runtimes. To load the full dataset, run all cells under the heading:

- Full Dataset - Binary Classification Only

To import data from the CSV files in a proportional manner to create the Reduced Dataset, run the following sections:

- Reduced Training Dataset Creation - Binary Classification
- Reduced Test Dataset Creation - Binary Classification

---

[5] How to read multiple data files into Pandas: https://www.geeksforgeeks.org/python/how-to-read-multiple-data-files-into-pandas/

[6] Python Glob: Filename Pattern Matching: https://pynative.com/python-glob/

[7] Stratified Sampling in Pandas: https://www.geeksforgeeks.org/python/stratified-sampling-in-pandas/

4.2.2.2 Categorical Classification

To import data from the CSV files in a proportional manner for the Reduced Dataset and apply the relevant category label, run the following sections:

- Reduced Training Dataset Creation - Binary Classification
- Reduced Test Dataset Creation - Binary Classification

5 Dataset Pre-Processing

Jupyter Notebook Section Name: *"Dataset Information and Pre-Processing"*

Code in this section is used to display information on the dataset and perform pre-processing steps in order to prepare the data for use with machine learning algorithms and hyperparameter tuning methods.

5.1 Display Dataset Information

This section displays the features contained within the dataset and shows how many instances are contained in the dataframes created after executing the code from the previous section.

5.2 Standardise Feature Names

Used to replace spaces with underscores, change all feature names to lowercase and correct a typo in a feature name.

5.3 Drop Duplicates, Check for Nulls

Drops any duplicate instances, and searches for any null ("np.nan") entries. Also checks for blanks, "?", or "–" characters and replaces them with nulls before repeating a check for nulls. Done for both train and test datasets.

5.4 Split Dataframes into X_train, y_train, X_test and y_test

This code splits both the training and test dataframes into separate dataframes with the input features (X_) and the target feature (y_), and this can be done for either binary or categorical classification, depending on the dataframe imported in section 3.2.2.

5.5 Fit and transform X_train dataset and transform X_test dataset without fitting

Standardisation is performed by scaling the X_train dataframe using the StandardScaler() function. The same scaling is then performed on the X_test dataframe without fitting the data, ensuring that the same scaling is used for both train and test features.

6 Feature Engineering

Jupyter Notebook Section Name: *"Feature Engineering Tasks"*

6.1 Variance Threshold Test - Drop Features with zero variance

Variance threshold testing is performed on the X_train dataframe. The "drate" column is identified as a feature with no variance, and is removed from the X_train and X_test datasets as a result.

6.2 Protocol Type Feature Assessment and Removal

The “protocol_type” feature was examined and found to not provide utility in this dataset, so this section of code drops this feature from the X_train and X_test datasets.

6.3 Further Reduce Dataset Sizes to 10% of Previously Reduced Dataset

Perform stratified sampling of both the train and test dataframes to improve processing times. The “train_size=0.1” and “test_size=0.1” variables in this section specify that 10% of the data will be retained. The “stratify=” variable defines that the data should be proportionately split based on the specified feature, which is y_train or y_test, the target feature. This ensures the balance of the dataset is unaffected by the split.

7 Classification Algorithms and Hyperparameter Tuning

Jupyter Notebook Section Name: *“Classification Algorithms and Hyperparameter Tuning”*

Each of the following machine learning algorithms has its own subsection within this section of the code:

- Decision Tree (DT)
- Random Forest (RF)
- Support Vector Machine (SVM)
- K-Nearest Neighbours (KNN)
- Multi-Layer Perceptron (MLP)

All of the above sections contain further subsections within in order to perform different tasks related to that algorithm,. Each of the subsections is detailed under the subheadings within section 6 of this report.

7.1 Hyperparameter Tuning Methods

For the Binary or Categorical Dataset imported in section 4.2.2, there are three methods of hyperparameter tuning that may be performed, detailed in the sections to follow. The input parameters used for each algorithm differ depending on the operation of the algorithm. The output format for each tuning method will be the same, and consists of the following information:

- The number of variables (candidates) specified, multiplied by the cross-validation specified, giving the total number of “fits” performed. (Note: For Bayesian Optimisation this information is in a different format. The entire list of iterations performed is output in real time as it is processed, with each iteration numbered.)
- The time taken (in seconds) for the hyperparameter tuning
- A list of the best performing parameters found, in the format of “ ’parameter’: value”

The format is shown in the screenshot below:

```
Fitting 4 folds for each of 24 candidates, totalling 96 fits
GridSearchCV Time: 155.22 seconds
Best performing parameters for the dataset are as follows:
{'criterion': 'gini', 'max_depth': 20, 'max_features': None, 'min_samples_leaf': 4, 'min_samples_split': 2}
```

Some common input parameters are shared by all three tuning methods used. The amount of cross validation to perform can be modified with the "cv=" parameter. The "scoring=" parameter should be set to 'f1' for binary classification and "f1_macro" for categorical classification, and either can be commented out using "#" as necessary.

Information on the basic use of the GridSearchCV and RandomizedSearchCV functions was obtained from a KDnuggets[8] web page. A Medium article which discussed Bayesian Optimization[9] was used as a starting point for the Bayesian Optimization hyperparameter tuning code implementation.

7.1.1 GridSearchCV Hyperparameter Tuning

GridSearchCV will search for the optimal hyperparameters from a defined set of input parameters (param_grid_<algorithm>), and will work through all possible combinations of the parameters defined, known as a grid search. "n_jobs=" controls how much processor usage the function can access, with a setting of -1 meaning the maximum possible available in order to maximise the speed of the tuning.

7.1.2 RandomizedSearchCV Hyperparameter Tuning

RandomizedSearchCV will attempt to find optimal hyperparameters from within the specified distribution of input parameters (param_dist_<algorithm>). Unlike GridSearchCV, it does not work through every possible combination of the input parameters, and instead selects them randomly from the input range for each parameter – a random search. The numpy linspace function is used to provide a randomly spaced sample of integers or floating point values (50 values by default) to choose from for each numeric parameter. Similar to GridSearchCV, "n_jobs" controls processor usage.

7.1.3 Bayesian Optimization Hyperparameter Tuning

Bayesian Optimization tries to locate the best set of hyperparameters in an algorithmic manner, using input parameters bound at lower and upper ends (param_bounds). It works iteratively in an effort to locate the highest performing parameters. There are two types of iterations. Firstly, it tries random values for the hyperparameters, and the number of attempts is specified with the "init_points=" parameter. Then, using the best approximation found using the initial searches as a starting point, it runs further iterations based on the value of the "n_iter=" setting.

7.2 Classification Tasks

Each algorithm selected for this research may be used to perform both binary or categorical classification, depending on the dataset loaded in section 4.2.2. The algorithm may be run with the default parameters (untuned), or with the parameters dictated by the output of any hyperparameter tuning method performed as described in Section 7.1. Certain parameter sets found using hyperparameter tuning can be set by uncommenting and commenting out code as desired. The algorithm will be trained against the training dataset, then predictions will be performed using the test dataset to assess performance.

---

[8] Hyperparameter Tuning: GridSearchCV and RandomizedSearchCV, Explained: https://www.kdnuggets.com/hyperparameter-tuning-gridsearchcv-and-randomizedsearchcv-explained

[9] Step-by-Step Guide: Bayesian Optimization with Random Forest: https://drlee.io/step-by-step-guide-bayesian-optimization-with-random-forest-fdc6f329db9c

The results output for each algorithm consists of the following:

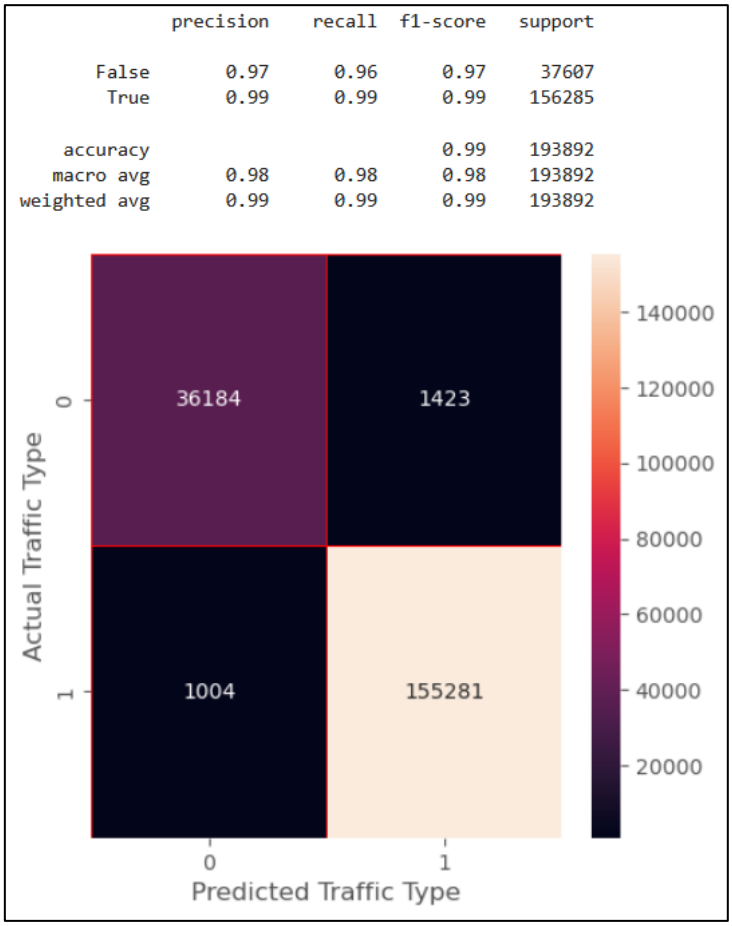
1. Results table consisting of various metrics detailed within the accompanying report. The format differs slightly for Binary and Categorical Classification methods, as shown in the subsequent sections.
2. A Confusion Matrix accompanied by a classification report. This provides a visual overview of the results and enables examination of true positives, true negatives, false positives and false negatives.
3. For informational purposes, the feature importances can be obtained for the DT and RF algorithms to show the most informative features used by the algorithm during its predictions. (Due to time constraints with running alternative methods of calculating the importance of features for the SVM, KNN and MLP algorithms, these were not carried out during the research.)

7.2.1 Binary Classification

For Binary Classification tasks, the table will have a single row of values, showing results as per the screenshot below. Accuracy, Precision, Recall and F1-score are all displayed as a percentage.

Training Time (s)	Inference Time (s)	Accuracy	Precision	Recall	F1
14.514	0.099	98.75	98.74	98.75	98.75

The next cell prints a classification report accompanied by a confusion matrix. For the confusion matrix, 0 indicates benign (malicious=0/False), while 1 (True) is malicious.



If using DT or RF, feature importances can be displayed to show the 20 most important features used for predictions by the algorithm. The screenshot below displays the format, showing only the top 5 as an example:

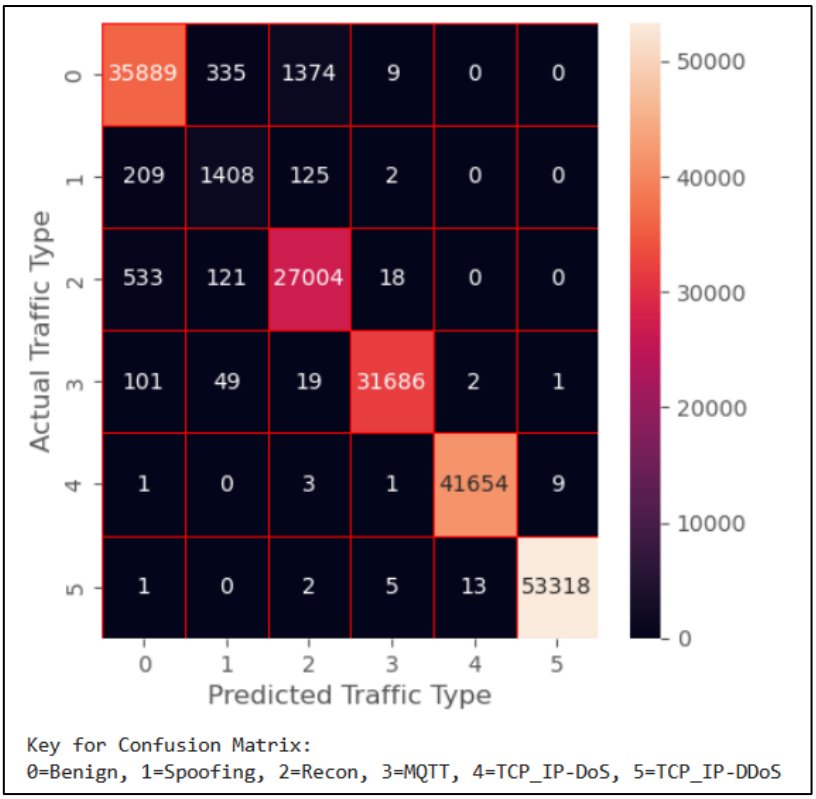
	feat_importance
rst_count	0.556360
psh_flag_number	0.110701
iat	0.098405
header_length	0.052992
rate	0.040243

7.2.2 Categorical Classification

Categorical Classification tasks will output a similar table with Accuracy, Precision, Recall and F1-score displayed as a percentage. The classification report will also be output when the first cell is executed, which includes the Precision, Recall and F1-score for each of the six categories, identified by an integer from 0-5. Overall macro and weighted averages can also be seen in this report.

```
  Training Time (s)    Inference Time (s)    Accuracy    Precision        Recall        F1
-------------------  --------------------  ----------  -----------  ------------  --------
             19.297                 0.171       98.49        98.52         98.49      98.5

              precision    recall  f1-score   support

           0     0.9770    0.9543    0.9655     37607
           1     0.7360    0.8073    0.7700      1744
           2     0.9466    0.9757    0.9609     27676
           3     0.9989    0.9946    0.9967     31858
           4     0.9996    0.9997    0.9997     41668
           5     0.9998    0.9996    0.9997     53339

    accuracy                         0.9849    193892
   macro avg     0.9430    0.9552    0.9488    193892
weighted avg     0.9852    0.9849    0.9850    193892
```

The second cell prints the confusion matrix, along with a key for mapping integer values to attack type categories. This allows quick visual analysis of which attack types are being classified correctly and which categories the algorithm is performing poorly when predicting.



Again if using the DT or RF algorithm, feature importances can be displayed in the same manner as done for Binary Classification:

	feat_importance
avg	0.351871
iat	0.350817
rate	0.103570
number	0.103133
header_length	0.054939

8 Shutting Down Jupyter Notebook

Once finished with code execution and results gathering, the Jupyter Notebook may be closed in order to ensure system resources are freed up successfully. To do this, simply close any browser windows related to Jupyter Notebook. Following this, select the Anaconda Powershell Prompt that is open and running Jupyter Notebook, and press Ctrl+C. A message similar to the one shown below will appear to confirm that the application has been stopped successfully, and the window may be closed.

```
[I 2025-07-11 14:51:58.122 ServerApp] Interrupted...
[IPKernelApp] WARNING | Parent appears to have exited, shutting down.
(iomtml) PS C:\Users\Charles>
```

References

Dadkhah, S., Neto, E.C.P., Ferreira, R., Molokwu, R.C., Sadeghi, S., Ghorbani, A.A. (2024) 'CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT', *Internet of Things*, 28, pp. 101351. https://doi.org/10.1016/j.iot.2024.101351