Configuration Manual

MSc Research Project
MSc in Data Analytics

Tejasvi Mirle Nataraja
Student ID: 23217120

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Tejasvi Mirle Nataraja		
Student ID:	23217120		
Programme:	Master of Science	**Year:**	2024-2025
Module:	Research Project		
Lecturer:	Vladimir Milosavljevic		
Submission Due Date:	January 29, 2025		
Project Title:	**Advanced thread detection in the IoT networks using hyper parameter tuned machine learning models**.		
Word Count:	**100**	**Page Count:**	6

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: Tejasvi Mirle Nataraja

Date: 29-01-2025

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

Tejasvi Mirle Nataraja
Student ID: 23217120

1 Environment setup

For executing the code, Google Colab IDE has been used with multiple runtime. For slow run of the code or binary classification CPU or free T4 GPU has been used. For intense classification tasks like multi class classification we have used paid subscription of colab pro has been used where 100 compute GPU units have been utilized which is described in Figure 2.

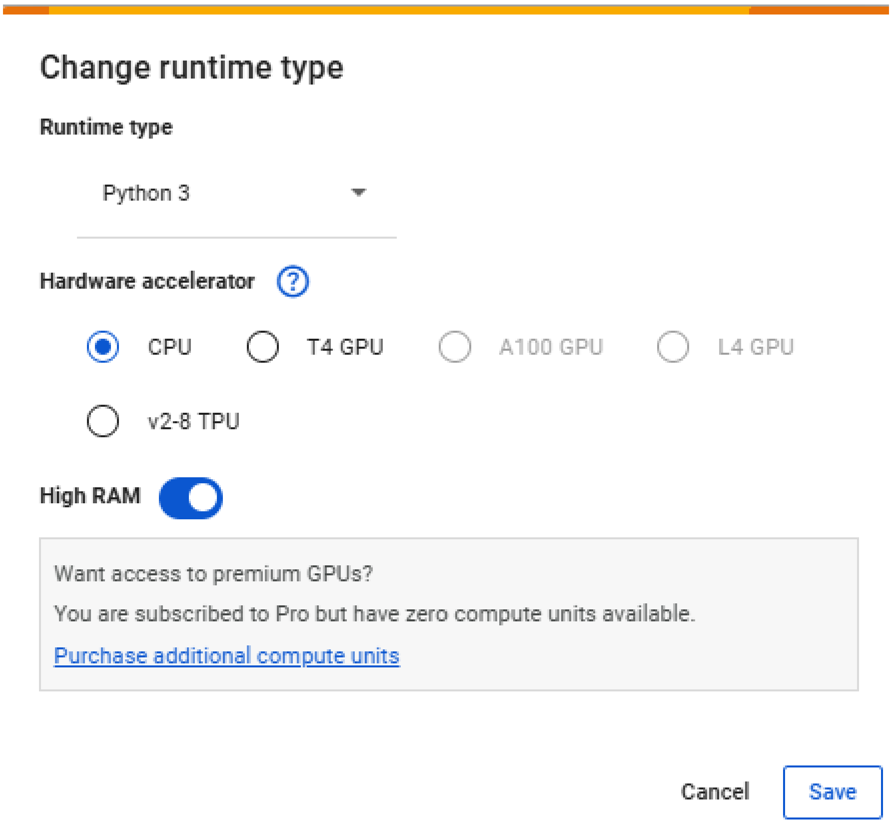

Figure 1: Runtime setup of Google Colab

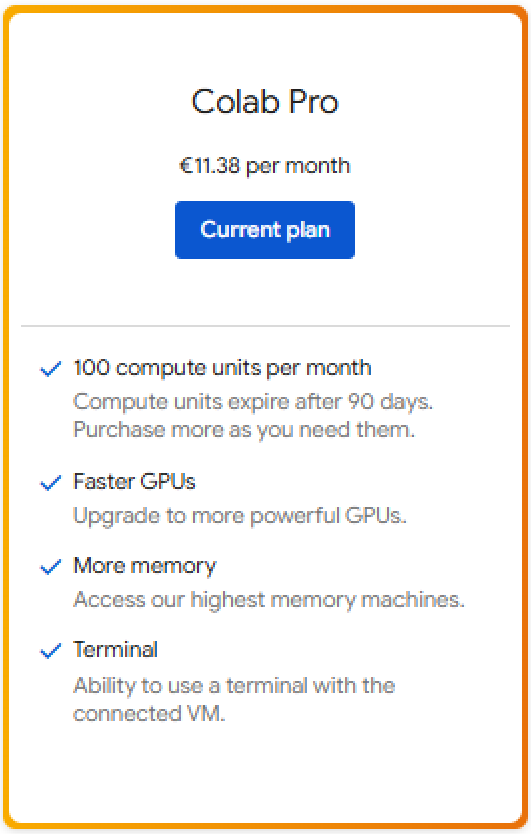

Figure 2: Paid subscription for 100 GPU units for multi class classification (A100GPU).

2 Package management

All necessary libraries has been imported available in python.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics         import accuracy_score, confusion_matrix, classification_report, precision_score, recall_score, f1_score
from sklearn.preprocessing   import StandardScaler
from sklearn.linear_model    import LogisticRegression
from sklearn.svm             import SVC
from sklearn.neighbors       import KNeighborsClassifier
from sklearn.ensemble        import RandomForestClassifier
from sklearn.neural_network  import MLPClassifier
from sklearn.naive_bayes     import GaussianNB, MultinomialNB, BernoulliNB
```

Figure 3: Code snippet of necessary libraries import

3 Random sampling

Library for random sampling.

```
from imblearn.under_sampling import RandomUnderSampler
```

Figure 4: Library for random sampling

4 Model Building

An example of Logistic regression model building process.

```
# Create an instance of the LogisticRegression model
clf = LogisticRegression()

default_params = clf.get_params()
print(f"Training model with default hyperparameters of: {default_params}")

# Fit the model to the training data
clf.fit(X_train_resampled, y_train_label_resampled)

# Predict the labels for the test data
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = clf.score(X_test, y_test_label)
print("Accuracy:", accuracy)

# save accuracy for later comparison
accuracy_lr_undersampled_unoptimized = accuracy

# show a running total of elapsed time for the entire notebook
show_elapsed_time()
```

Figure 5: Model building code

5 Hyperparameter tuning

Hyper parameter tuning done to improve the model performance.

```
# Create an instance of GridSearchCV
grid_search = GridSearchCV(clf, param_grid, cv=cv_count, n_jobs=-1)

# Fit the grid search to the training data
grid_search.fit(X_train_resampled, y_train_label_resampled)

# Get the best hyperparameters
best_params = grid_search.best_params_
best_scores = grid_search.best_score_
print("Best Parameters:", best_params)
print("Best Scores:", best_scores)

# Create a new instance of the model with the best hyperparameters
clf = LogisticRegression(**best_params)

# Fit the model to the training data
clf.fit(X_train_resampled, y_train_label_resampled)

# Predict the labels for the test data
y_pred = clf.predict(X_test)
```

Figure 6: A glimpse of hyper parameter tuning

6 Results:

All the results are tabulated in an excel file which is attached as part of code artefacts.

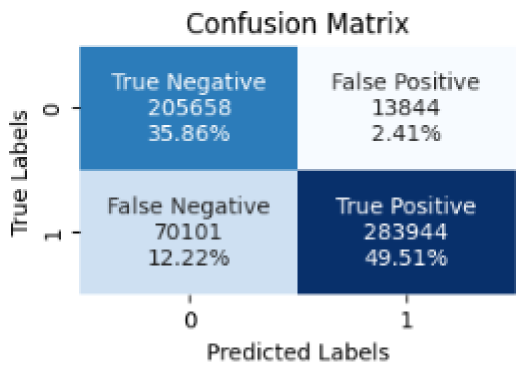

Figure 7: Confusion matrix of logistic regression model before hyper parameter tuning for binary classification.

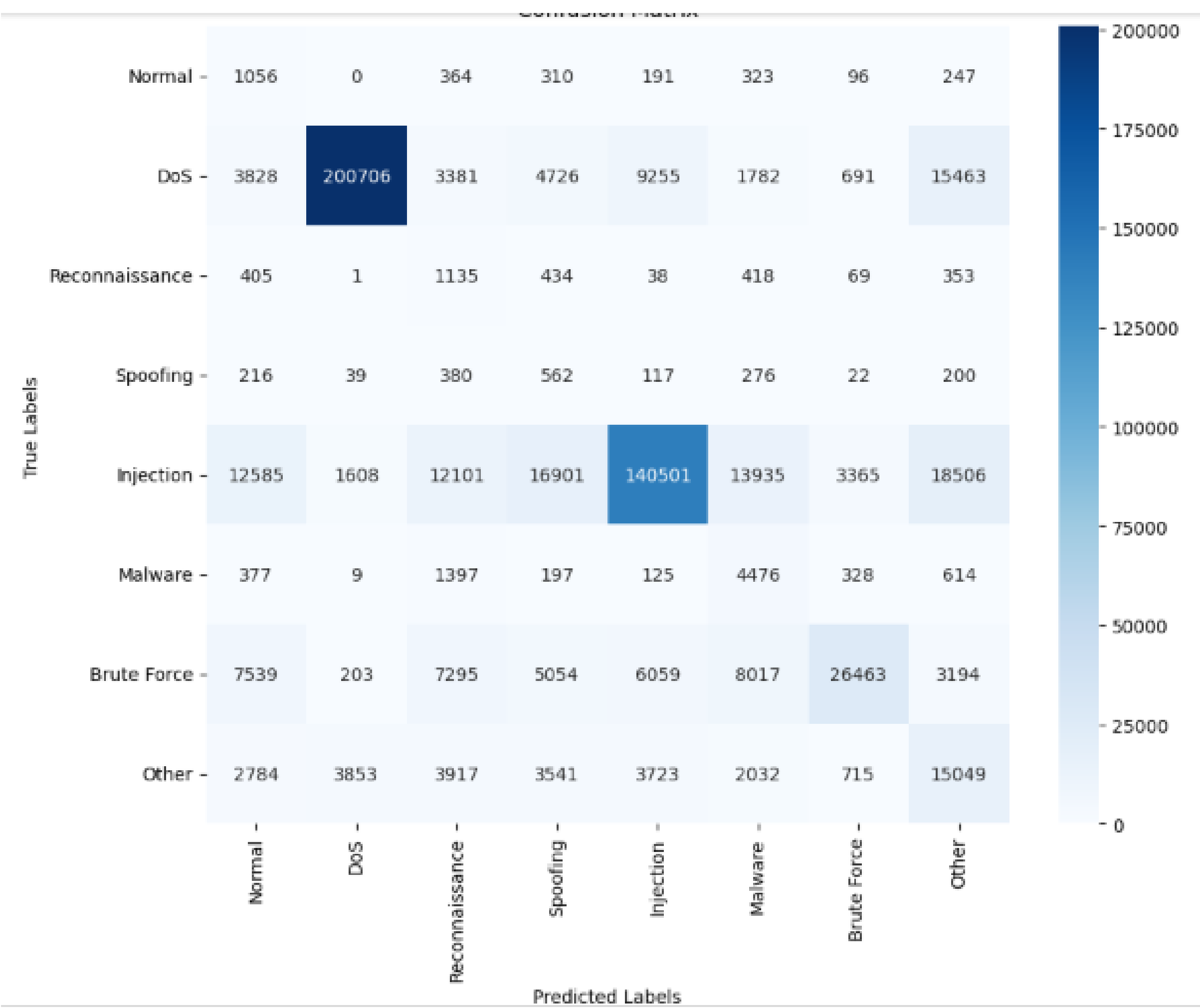

Figure 8: Confusion matrix of logistic regression model before hyper parameter tuning for multiclass classification.

7 Label encoding for multi class classification

For multi classification, initially the attack types are mapped to particular category of 8 types in general and later label encode.

```python
# Define mapping from specific attack types to general categories
attack_mapping = {
    'BenignTraffic': 'Normal',
    'Mirai-udpplain': 'DoS',
    'MITM-ArpSpoofing': 'DoS',
    'Recon-PortScan': 'Reconnaissance',
    'DNS_Spoofing': 'Spoofing',
    'Recon-OSScan': 'Reconnaissance',
    'XSS': 'Injection',
    'Recon-HostDiscovery': 'Reconnaissance',
    'CommandInjection': 'Injection',
    'VulnerabilityScan': 'Other',
    'Backdoor_Malware': 'Malware',
    'BrowserHijacking': 'Malware',
    'DictionaryBruteForce': 'Brute Force',
    'SqlInjection': 'Injection',
    'Recon-PingSweep': 'Reconnaissance',
    'Uploading_Attack': 'Other'
}

# Apply the mapping
df['Attack_category'] = df['Attack_type'].map(attack_mapping)
```

Figure 9: Mapping of IoT attack types to 8 categories

8 References:

1. Scikit-learn , *Scikit-learn: Machine Learning in Python*. Available at: https://scikit-learn.org/stable/ (Accessed: 8 September 2024)
2. Matplotlib , *Matplotlib 3.10 Documentation*. Available at: https://matplotlib.org/stable/index.html (Accessed: 6 September 2024).
3. Seaborn , *Seaborn: Statistical Data Visualization*. Available at: https://seaborn.pydata.org/ (Accessed: 5 September 2024).
4. Pandas, *Pandas Documentation*. Available at: https://pandas.pydata.org/docs/ (Accessed: 5 September 2024)
5. NumPy , *NumPy Documentation*. Available at: https://numpy.org/doc/stable/ (Accessed: 5 September 2024).