National College of Ireland

Configuration Manual

MSc Research Project
Cyber security

Ganeshkumar Prabakaran
Student ID:23410311

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Ganeshkumar Prabakaran

Student ID: 23410311

Programme: MSc Cybersecurity **Year:** 2026

Module: Practicum Part 2

Supervisor: Michael Prior

Submission Due Date: 28th January 2026

Project Title: Developing Efficient Machine Learning Algorithms for Anomaly Detection in Network Traffic for Cybersecurity

Word Count: 922 **Page Count:** 5

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: Ganeshkumar Prabakaran

Date: 28th January 2026

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

Ganeshkumar Prabakaran
Student ID: 23410311

1. Overview

This configuration manual describes all essential software tools, libraries, dataset files, Python dependencies, and experiment settings required to successfully reproduce the machine-learning anomaly detection experiments carried out using the NSL-KDD (KDDTrain+) dataset. It outlines the required environment, dataset preparation steps, preprocessing configuration, model hyperparameters, and output generation settings. No installation instructions for standard software are included.

2. Required Software Tools & Versions

The experiments were executed in a Python environment configured with the following components:

2.1 Programming Language

- **Python Version:** 3.8+ or 3.9+

2.2 Required Python Libraries

The following libraries must be installed in the environment. Exact versions are not required, but the latest stable releases are recommended:

Library	Purpose
NumPy	Numerical operations
Pandas	Data handling and feature processing
SciPy	Additional utilities (ARFF support, not used directly)
Scikit-learn	Preprocessing, ML models, evaluation
Matplotlib	Data visualisation
Warnings (stdlib)	Ignore warnings

2.3 Required Dataset Files

- **KDDTrain+.txt** (from NSL-KDD dataset)
- File must be located in the **same directory** as the Python script or notebook.

2.4 Execution Environment

- Any Jupyter Notebook environment, VS Code, PyCharm, or command-line Python REPL
- Hardware requirements:

- **RAM:** 8 GB minimum (due to one-hot encoding expansion)
- **CPU:** Multi-core recommended (Random Forest uses parallel processing)

3. Dataset Configuration

3.1 Dataset Structure

KDDTrain+.txt must contain all 43 NSL-KDD features including:

- 41 traffic features
- label
- difficulty

Column names are manually assigned in the script.

3.2 Column Name Mapping

The following list is applied during configuration:

```
col_names = [
'duration','protocol_type','service','flag','src_bytes','dst_bytes','land','wrong_fragment',
'urgent','hot','num_failed_logins','logged_in','num_compromised','root_shell','su_attempted',
'num_root','num_file_creations','num_shells','num_access_files','num_outbound_cmds',
'is_host_login','is_guest_login','count','srv_count','serror_rate','srv_serror_rate',
'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','srv_diff_host_rate',
'dst_host_count','dst_host_srv_count','dst_host_same_srv_rate','dst_host_diff_srv_rate',
'dst_host_same_src_port_rate','dst_host_srv_diff_host_rate','dst_host_serror_rate',
'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rerror_rate','label','difficulty'
]
```

3.3 Required Dataset Preprocessing Steps

The following preprocessing must be applied:

1. **Feature Type Handling**
   - Identify categorical: protocol_type, service, flag.
   - Numerical: All other numeric fields except difficulty.
2. **Service Reduction**
   - Top 20 frequent services retained.
   - Others grouped into "other" category.
3. **Categorical Encoding**
   - One-hot encoding for categorical variables.
   - No drop of first category to ensure full feature representation.

4. **Numerical Preprocessing**
    - Missing values imputed using **median strategy**.
    - Standardisation using **StandardScaler**.
5. **Label Transformation** ○ Convert all labels not equal to "normal" into "attack".
    - Encode into binary (normal = 1, attack = 0).
6. **Final Feature Matrix** ○ Concatenate scaled numeric features + one-hot encoded categorical features.

4. Train-Test Split Configuration

The following split configuration must be used to ensure identical results:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)
```

Important settings:

- **Test size:** 25%
- **Random state:** 42
- **Stratification:** Enabled, ensures same class ratio in train & test sets

5. Model Configuration

This section defines all hyperparameters used for each ML model.

5.1 Decision Tree Classifier

```
DecisionTreeClassifier(     random_state=42,     max_depth=12
)
```

5.2 Random Forest Classifier

```
RandomForestClassifier(     n_estimators=200,
random_state=42,     n_jobs=-1
)
```

Parallel processing allowed (n_jobs=-1)

Ensemble model chosen for best performance

5.3 K-Means Clustering

```
KMeans(
    n_clusters=2,
```

```
    random_state=42,     n_init=10
)
```

Two clusters: normal vs attack

Full dataset used for clustering

6. Evaluation Configuration

6.1 Metrics Used

- Accuracy
- Precision
- Recall
- F1-score
- ROC-AUC (supervised & unsupervised)
- Classification Report
- Confusion Matrix
- Purity Score for K-Means

6.2 ROC-AUC

- For Random Forest: predict_proba used
- For K-Means: Distance-based anomaly score used

6.3 Purity Score Formula

```
purity = sum(max(count of each class per cluster)) / total samples
```

7. Visualisation Configuration Enabled Plots

- Class distribution bar chart
- Protocol distribution bar chart
- Top 10 service distribution
- Flag distribution
- Correlation heatmap
- Confusion matrices
- ROC curves

Plot Characteristics

- Matplotlib 10x4, 6x4, 12x10, and 6x5 inch figures
- "coolwarm" colormap for heatmap
- Grid enabled for ROC curve

8. Execution Order

To successfully replicate results, steps must be executed in this exact order:

1. Import all libraries

2. Load dataset
3. Apply column names
4. Perform EDA (optional but recommended)
5. Apply preprocessing (encoding + scaling)
6. Prepare final feature matrix
7. Split into train/test sets
8. Train and evaluate Decision Tree
9. Train and evaluate Random Forest
10. Extract Random Forest feature importance
11. Run K-Means clustering
12. Compute purity score
13. Generate ROC curve for K-Means