COS30049 - Computing Technology Innovation Project

Assignment 2

# Network Traffic Classification for Anomaly Detection

Session 23 – Group 3 (Tutor – Ricky Dong)

|  |  |  |
| --- | --- | --- |
| No. | Student Name | Student ID |
| 1. | Nguyen Duc Thang | 104776473 |
| 2. | Chann Muninth Khun | 105249293 |
| 3. | Soh Chun Jie | 105062092 |

**Table of Contents**

[Network Traffic Classification for Anomaly Detection 1](#_Toc210790865)

[1. Introduction 1](#_Toc210790866)

[2. Problem Framing 1](#_Toc210790867)

[3. Data Collection 2](#_Toc210790868)

[3.1.1 UNSW-NB15 2](#_Toc210790869)

[3.1.2 Ip\_addresses\_sample 4](#_Toc210790870)

[3.1.3 Basic Dataset (4Network) 6](#_Toc210790871)

[3.2 Challenges faced 8](#_Toc210790872)

[4. Data Processing 8](#_Toc210790873)

[4.1 Preprocessing & Cleaning 8](#_Toc210790874)

[4.1.1 unsw-nb15 preprocessing & cleaning 8](#_Toc210790875)

[4.1.2 ip\_address\_sample preprocessing & cleaning 9](#_Toc210790876)

[4.1.3 basic dataset preprocessing & cleaning 10](#_Toc210790877)

[4.2 Schema harmonisation & merge 11](#_Toc210790878)

[4.3 feature engineering 11](#_Toc210790879)

[4.4 handling missing values, encoding & scaling 12](#_Toc210790880)

[4.5 handling data imbalance 12](#_Toc210790881)

[4.6 outputs 13](#_Toc210790882)

[4.6.1 Feature Correlation analysis 13](#_Toc210790883)

[4.6.2 Dimensionality reduction and visualization 14](#_Toc210790884)

[5. Machine Learning Model Selection 16](#_Toc210790885)

[6. Implementation 16](#_Toc210790886)

[6.1 Technical Implementation 16](#_Toc210790887)

[6.2 Implementation Evaluation 16](#_Toc210790888)

[7. Conclusion 16](#_Toc210790889)

[8. Bibliography 16](#_Toc210790890)

[9. Appendix 16](#_Toc210790891)

# 1. Introduction

This project is motivated by the crucial requirement to proactively predict and mitigate various cyber threats within network traffic. Traditional security systems frequently rely on predefined signatures which make it vulnerable to innovative and evolving attacks. To address this problem, our project will develop a comprehensive anomaly detection system by comparing three different machine learning models such as supervised classification, unsupervised clustering, and time-series forecasting to find out the most effective models to identify abnormal activity from normal behaviour.

The intended users of this system include government cybersecurity agencies, cloud service providers, and large enterprises whose manage vast amounts of sensitive data that are the primary targets for attackers. By identifying which machine learning models are the most effective for anomaly detection, our project ensures a more reliable system for security professionals to strengthen network resilience and combat an increasingly complex threat landscape.

# 2. Problem Framing

Network intrusion detection is a recognized but continually growing problem in the field of cybersecurity. The specific problem is to identify between normal and malicious traffic from real-world where the data is typically inconsistent, dimensional, and updating frequently. In this kind of environments, signature-based methods became difficult in identifying new or unfamiliar attacks, and the rare types will be easily overwhelmed by majority traffic attacks such as DoS. Thereby, our project aims to address this kind of problems by applying and comparing different machine learning models to get the most suitable model that contained highly adaption to changing patterns while maintaining its accuracy and without overwhelming analysts with false positives.

However, existing network detection solutions still face several important limitations including:

1. High false-positive rates

Since network traffic is continually changing based on user activity, traffic congestion, and scheduled data transfers, traditional intrusion detection systems frequently misidentify normal behaviour as malicious. Their reliance on predefined rules and criteria results in many false alerts which will overwhelm analysts and reducing the trust in the system.

1. Data imbalance

In most of the real-world datasets, normal traffic known as the majority of data while specific attack types only happen rarely. This imbalance makes the traditional detection models become unreliable, allowing them to recognize typical behaviours but fail to detect unusual and serious threats. For example, attacks like “User-to-Root” (U2R) or “Remote-to-Local” (R2L) often get undetected or misidentified as normal behaviour.

1. Feature overlap

Since many network traffic share similar features such as packet size, duration or byte count between normal and malicious traffic. This makes it hard for basic detection systems to identify between the two traffic and allow attackers take advantage by making their traffic look like normal behaviours thereby making the detection even more difficult. These overlapping patterns cause in incorrect predictions and poor detection performance without having more advanced models that can recognize small distinctions between the normal and malicious traffic.

Machine learning provides an effective solution to these limitations by learning from data rather than static rules or signatures. It can adapt to changing traffic patterns, manage imbalanced datasets using techniques like resampling or class weighting, and identify hidden relationships between overlapping features. This project will use and compare supervised classification, unsupervised clustering, and time-series forecasting models to determine the most effective and adaptable models for detecting anomalies while maintaining the scalability and minimizing false positives.

# 3. Data Collection

Our dataset was compiled from three valuable sources to enable both large attack coverage and depth of historical behaviour for anomaly detection

### 3.1.1 UNSW-NB15

We chose UNSW-NB15 as the supervised backbone because it provides comprehensive modern flow-level data with 49 security-relevant features and 10 labels (9 distinct attack categories and normal traffic) which gave broad coverage for both binary and multiclass experiments. In our analysis for this dataset, we used pandas to load the four CSV shards, verified the schema and normalised the column names to ensure the consistency. We also identified class balance using value\_counts () and visualized distributions using Seaborn and Matplotlib for normal vs attacks and each attack’s family counts. The analysis results (shown in Figure 1 - 3) demonstrates the large scale of this dataset and the significant class imbalance issue which normal traffic takes priority while Generic is significantly larger while Shellcode/Worms are extremely rare within attacks. A notable missing values around 5 million cells across several features also been flagged which will be addressed in future analysis and model development. These observations motivated stratified splits, class weighting, further tuning, and missing-value handling in later.

Source: https://research.unsw.edu.au/projects/unsw-nb15-dataset

A screen shot of a computer

AI-generated content may be incorrect.

Figure 1. UNSW-NB15 DataFrame Info

A screenshot of a graph

AI-generated content may be incorrect.

Figure 2. Distribution of attack categories in UNSW-NB15

A graph of a bar graph

AI-generated content may be incorrect.

Figure 3. Distribution of Normal vs. Attack labels in UNSW-NB15

### 3.1.2 Ip\_addresses\_sample

We included ip\_addresses\_sample dataset to capture long-term behaviour that per-flow datasets are unable to. As shown in Figure 4, the table contains complete, non-null data for traffic quantities (flows, packets, bytes), diversity metrics (destination ASNs/ports/IPs and their averages/standard deviations), protocol ratios (TCP/UDP), direction ratios (packets/bytes), average duration, and average time-to-live (avg\_ttl). We used pandas for loading and inspection then using Matplotlib and Seaborn for time-series visualization to show bytes and packets per day (Figure 5) as well as daily flow counts (Figure 6). The figures demonstrate significant burstiness in bytes, relative stability in packet levels, and inconsistent flow activity with maximum over 3000 early in the execution before settling into lower-variance ranges. These patterns suggest periodic or event-driven behaviour and serve as a realistic basis for baseline or seasonality modelling and future anomaly detection such as identifying abnormal instances using continuous data.

Source: https://zenodo.org/records/13382427

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4. IP Address Sample Dataset Info

A graph showing a graph

AI-generated content may be incorrect.

Figure 5. Daily Bytes and Packets count within 280 days

A graph showing a green line

AI-generated content may be incorrect.

Figure 6. Daily Number of Network Flows within 280 days

### 3.1.3 Basic Dataset (4Network)

We used the provided basic dataset for quick prototyping and pipeline integrity checks since it provides interpretable, simple features (protocol, service, flags, src/dst bytes, simple counts) with classic anomaly detection labels (Normal, DoS, Probe, R2L, U2R). We analysed class balance and main attack types using pandas for loading and inspection, then using Seaborn and Matplotlib for visualization just like previously did for other datasets. The results show highly imbalance label space where Normal and DoS dominate, Probe follows, while R2L and U2R attacks are extremely rare.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7. Basic Dataset Info

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8. Category Distribution in Basic Dataset

A graph with multiple colored squares

AI-generated content may be incorrect.

Figure 9. Distribution of Traffic Categories in Basic Dataset

A screenshot of a computer

AI-generated content may be incorrect.

Figure 10. Top 15 Attack Types in Basic Dataset

## 3.2 Challenges faced

During data collection, we encountered several challenges as below:

1. Scale & access

Dataset like UNSW-NB15 has several GB shards which making loading and plotting memory-intensive. To address it, we downloaded the official shards, then used staged/chunked ingestion with pandas and limited columns for early EDA to control memory and rendering time.

1. Data Imbalance

Labels are significantly imbalance such as Normal and Generic are numerous while Shellcode/Worms/R2L/U2R are slightly rare. For this, we quantified and visualised the imbalance and planned layered splits, category-aware metrics, and class weighting for future models

1. Missing values

UNSW-NB15 has significant missing values which might affect the statistics and break models. To address this, we flagged the affected columns during collection and prepared a handling plan to overcome it.

To ensure the processed datasets are reliable for further modelling and evaluation, we will merge these datasets and talk about how to overcome the challenges we found in detail within the next section.

# 4. Data Processing

Our data processing pipeline combined three of the datasets discussed into a single, model-ready dataset using a leakage-free, repeatable methodology. We started by importing the initial CSVs, standardizing column names, enforcing valid data types and ranges, and removing duplicates. After that, we mapped each dataset onto a shared flow schema and combine it with a reliable label. Next, we created rate and density features which handled missing values with median or mode replacement, and processed inputs using one-hot encoding for categorical and standard scaling for numeric. At the end, we used a layered 80/20 train-test split, pre-processed just the training set to avoid leakage, used SMOTE to the training data to solve data imbalance, and stored all processed splits and transformers for future modelling.

## 4.1 Preprocessing & Cleaning

### 4.1.1 unsw-nb15 preprocessing & cleaning

We loaded the official splits and combined them, then normalized the columns by removing whitespace, lower-casing, and replacing spaces with underscores. We mapped the original attack\_cat into a simplified classification of five categories which is Normal, DoS, Probe, R2L, and U2R with an alternative of Other while the potential missing values will be classified as Normal. We then generated a consistent multiclass label from the mapped attack\_cat and a binary target where 0 for normal and 1 for any attack. This stage creates a clear, harmonised target space while retaining documentation of the original categories for review.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 11. Code for mapping and label generation

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12. Processed UNSW-NB25 DataFrame Head

### 4.1.2 ip\_address\_sample preprocessing & cleaning

For this dataset, we extracted the host identifier (id) from each filename and combined them into a single table. Because the file does not include IP addresses, we did not use it for joins; instead, we used id as the unique key. Since this dataset has no attack labels, we set category = ‘Normal’, binary\_label = 0, and attack\_cat = ‘normal’ for consistency. In order to make the time-aggregated data comparable to flow-style features, we produced stable summaries for each host by grouping with id and applying mean, standard, and maximum to accessible counters like the n\_bytes, n\_packets, n\_flows, and mean for avg\_duration. The multi-index columns were then flattened, and the auto-generated id name was normalized back to id. This results in a clean, host-level feature table with consistent labels and schema.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 13. Code aggregation and label assignment

A black screen with white text

AI-generated content may be incorrect.

Figure 14. Processed and Aggregated IP\_Address\_Sample DataFrame head

### 4.1.3 basic dataset preprocessing & cleaning

For this basic dataset, we just loaded the two files and left-joined them using the shared label key to recover descriptive attack categories for each record. We then renamed some labels (such as src\_bytes to sbytes) to make it consistent with another 2 datasets and other headers remained normalized and counters as numeric. We normalized the category text and created a consistent binary target which show 0 for “normal” and 1 for others. This provides a clean table ready for schema harmonisation and merge.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 15. Processed Basic DataFrame head

## 4.2 Schema harmonisation & merge

After cleaning and preprocessing all the datasets, we mapped them onto a shared flow schema and merged them into a single model-ready dataset using Pandas via row-wise concatenation while maintaining label consistency by harmonizing the data frame to a standard columns with header normalization and name alignments performed internally. When a dataset lacked a direct field, we used documented placeholders or 0 so future preprocessing could handle them consistently.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 16. Code on merging 3 harmonised DataFrame

A screenshot of a computer

AI-generated content may be incorrect.

Figure 17. Merged dataset preview

## 4.3 feature engineering

After we had a merged dataset, we created a small set of new features such as packet-rate measures for each direction (srate, drate), size of collection (total\_bytes, total\_pkts), and a packing-density indicator (bytes\_per\_pkt) with a small constant epsilon to avoid division-by-zero. These features capture temporal intensity, traffic quantity, and payload compactness which significantly improve the identification between normal and attack behaviour.

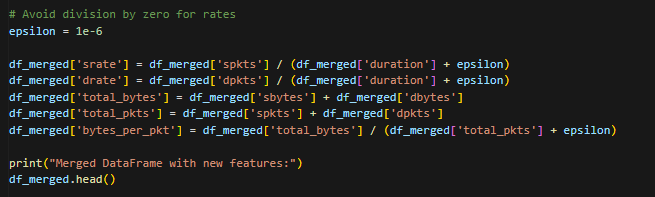


Figure 18. Code Snippet for Feature Development on the Merged Flow Schema

## 4.4 handling missing values, encoding & scaling

In order to handle missing values, a single scikit-learn preprocessing pipeline has been used to ensure that the process was reproducible and leak-free. Numerical features were attributed using the median, then standardized to zero mean and unit variance. Besides, categorical features were added with the most-frequent value and one-hot encoded with handle\_unknown = ‘ignore’ to ensure that unknown categories during testing did not break the model. Any structural gaps generated during schema harmonisation (such as features missing from a dataset and replaced with placeholders) will be handle consistently using the same imputation phase. The pipeline was only fitted with the training split and then used for both training and testing thereby limiting data leakage and producing a single, model-ready feature matrix with extended (OHE) features names.

A computer screen shot of code

AI-generated content may be incorrect.

Figure 19. Code Snippet of Missing-value Handling and Scaling via Scikit-learn Pipelines

## 4.5 handling data imbalance

To address the challenge of data imbalance, we first implemented a stratified 80/20 train-test split to maintain class proportions in the remaining dataset, then used SMOTE only ono the training data to synthetically overload minority classes while leaving the majority classes and the entire test set unchanged. We set the random\_state to 42 for repeatability, verified the impact by recording before and after class counts. Figure 21 shows the shift from an imbalanced distribution to a perfectly balanced training set with around 20% of each class after applied SMOTE.

A computer screen with white text

AI-generated content may be incorrect.

Figure 20. Code Snippet for SMOTE

**A green and red bar graph

AI-generated content may be incorrect.**

Figure 21. Class Distribution before & after SMOTE

## 4.6 outputs

At the end we have provide a single, model-ready dataset together with repeatable preprocessing results and data monitoring visualizations. The dataset includes the harmonised flow schema, as well as created features and consistent labelling. To demonstrate data quality and structure, we have done the feature correlation analysis and dimensionality reduction visualizations to enable transparent reviewing, hyperparameter adjustment, and future model selection.

### 4.6.1 Feature Correlation analysis

We calculated correlations on the processed numeric feature set and displayed the most associated features. As predicted, the result (Figure 22) demonstrates that volume features move together strongly while protocol indicators are mainly opposite.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 22. Feature Correlation Heatmap

### 4.6.2 Dimensionality reduction and visualization

We analysed internal structure using PCA and t-SNE on pre-processed training data. The result of PCA (Figure 23) shows a gradual accumulation of variance which demonstrating that variance is distributed across multiple dimension which is useful for setting expectations for linear compression and motivating non-linear models.

A graph of a number of concomitance

AI-generated content may be incorrect.

Figure 23. PCA Cumulative Explained Variance

t-SNE (Figure 24) demonstrates cohesive categories with significant separation for categories such as DoS and R2L, although Probe largely overlaps with other attack types which indicating behavioural similarities. Following preprocessing and data rebalancing, a binary t-SNE (Figure 25) shows a clearer separation between the Normal and Attack.

A screen shot of a computer generated image

AI-generated content may be incorrect.

Figure 24. t-SNE Project of Processed Training Data (Multiclass)

A blue and orange dots

AI-generated content may be incorrect.

Figure 25. t-SNE Project of Processed Training Data (Binary)

These graphs demonstrate that the designed features and preprocessing pipeline produce structure that models may utilize while also demonstrates where decision of models selection may be challenging.

# 5. Machine Learning Model Selection

# 6. Implementation

## 6.1 Technical Implementation

## 6.2 Implementation Evaluation

# 7. Conclusion

# 8. Bibliography

# 9. Appendix