Phase 2 Machine learning Report

Team no. : 1

Section : 1

Team Members: Ahmed Mohsen 202301138 , Jana Ahmad 202301800 , Ali Elrouby 202301680 , Mazen Ahmed 202300900

Problem Statement:

Cybersecurity threats are becoming increasingly diverse and sophisticated, especially in large-scale and high-traffic environments. This requires advanced machine learning solutions capable of identifying not just the presence of an attack (like the last phase), but the specific type of attack, to support faster and more effective incident response.

We built a system that can classify network traffic into multiple categories, including normal behavior and various forms of malicious activity such as DDoS, DoS, brute-force, and scanning attacks. The dataset used includes labeled examples of real-world traffic patterns extracted from packet flow features, allowing the model to learn and distinguish between normal and several attack types.

Challenges:

- Multiclass Imbalance:
  The dataset contains a highly imbalanced distribution of classes, with some attack types (like DDoS) appearing far more frequently than others. This imbalance poses a challenge for model fairness and performance on underrepresented classes.

- Feature Complexity:
  The dataset includes 21 numerical and binary features describing aspects such as packet duration, source/destination rates, protocol flags, and connection behavior. These features vary in scale and importance, requiring preprocessing, normalization, and feature selection.

Data Exploration:

Firstly, we began to explore the data, and found that there are 21 numerical features, and a label column that have 6 different classifications (DDoS, DoS, BenignTraffic, Mirai, Recon, MITM).

We applied .describe() to all numerical columns to summarize key statistics such as:

- **Mean, median, and standard deviation** of features like packet_duration, src_rate, and flow_time

- **Minimum and maximum values** to detect possible outliers or anomalies

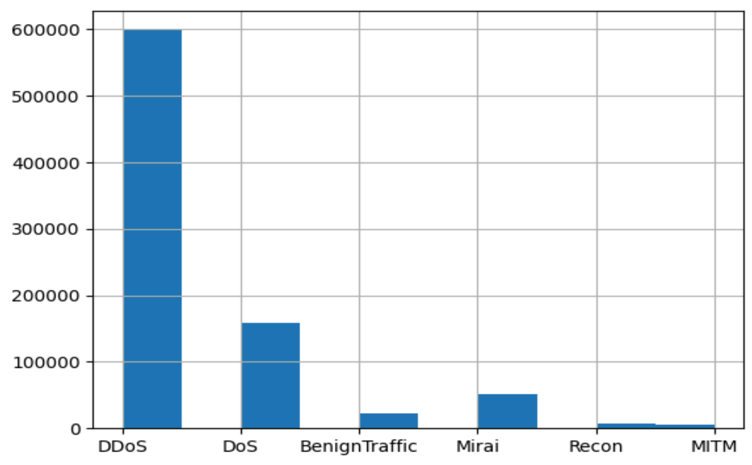
- **Skewness and scale** differences across features, prompting us to apply log scaling during preprocessing

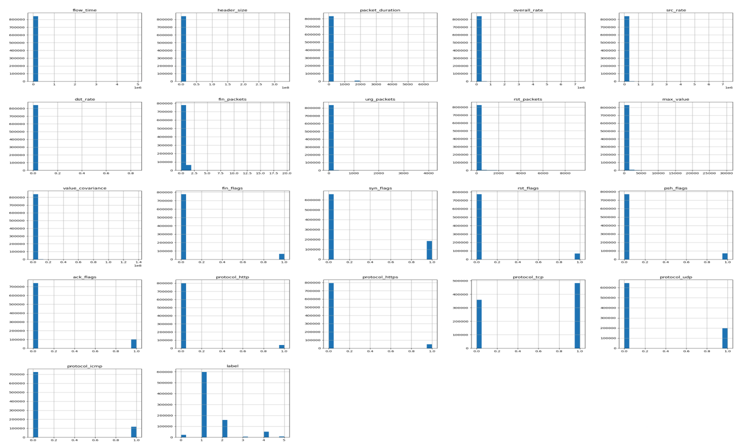
We checked for null or missing values and found none. We also identified and removed a duplicate column (overall_rate), which was identical to src_rate.

We visualized the distribution by a histogram of:

- Each numerical feature to identify skewed distributions and extreme values
- The target variable (label) to assess class distribution

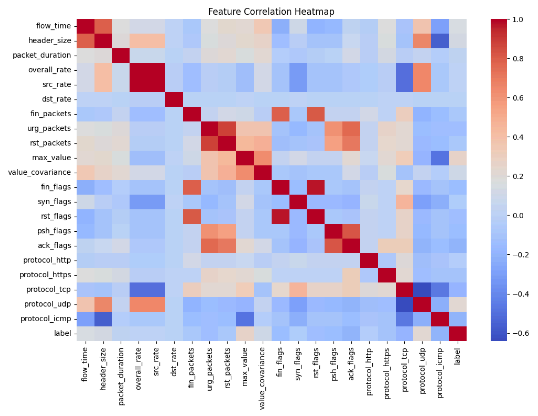
This revealed a significant class imbalance, with certain attack types (e.g., DDoS) dominating the dataset while others appeared infrequently.





Also, we explored feature correlations using correlation matrices to:

- Detect redundant or highly correlated features
- Understand relationships between rate-based metrics, timing, and flags



<u>Data Preprocessing:</u>

Many of the numerical features, such as packet_duration, src_rate, and flow_time, contained extremely large or skewed values. To reduce the impact of outliers and compress wide-ranging distributions, we applied logarithmic scaling using log1p (i.e., log(1 + x)) to all relevant numerical columns. This transformation improved model learning stability and convergence.

Additionally, we identified and removed the column overall_rate, which was found to be an exact duplicate of src_rate. Keeping both would introduce unnecessary redundancy and could bias the model.

We checked the dataset for identical rows and removed any detected duplicates. This step helped reduce overfitting and ensured that the model learned from unique patterns rather than memorized data.

To enhance the dataset's predictive power, we created several new features based on domain understanding and correlations observed during exploration. These included:

- **Derived rate features** (e.g., ratio of src_rate to dst_rate)
- **Interaction terms** combining protocol flags and connection timing
- **Composite indicators** capturing suspicious behavior patterns

These additional features aimed to capture more complex relationships and improve the model's ability to differentiate between attack types.

And to train the model, the target variable (label) was converted from categorical class names to numeric format using label encoding. This was necessary to train classifiers that expect numerical target variables, especially in multiclass setups.

Feature engineering:

After a lot of experiments, we came up with new features from the existing features that are already in the datasets. The new features are rate ratio, rate sum, flow time per byte, duration per rate, total flags, is flag heavy, is tcp heavy, is udp slow, rate diff, flow per header, is short packets, is one way.

Their equations are:

- ➢ Rate ratio = src rate / (dst rate + 1e-5)
- ➢ Flow time per byte = flow time/ (header size = -1)
- ➢ Duration per rate = packet duration/ (src rate + 1)
- ➢ Total flags = sum(ack_flags, fin_flags, syn_flags, psh_flags, rst_flags)
- ➢ Is flag heavy = (total flags>2)
- ➢ Is tcp heavy = protocol tcp * src rate
- ➢ Is udp slow = protocol udp * packet duration
- ➢ Rate diff = abs(src rate - dst rate)
- ➢ Flow per header = flow time / (header size + 1)
- ➢ Is short packet = (packet duration<10)
- ➢ Is one way = (src rate > 0) & (dst rate ==0)

<u>Model Selection and Implementation:</u>

We tried a lot of models, like:

- KNN
- LogisticRegression (One vs Rest)
- SVM
- Decision tree
- Random forest
- Neural networks (MLPClassifier)
- XGBoost
- LightGBM
- Catboost

And we tried two ensemble techniques, soft voting and stacking. With also tuning the models with different hyperparameters and trying to combine them in a lot of possible combinations to find the best accuracy on test_data.

For tuning the hyperparameters, we tried a lot of techniques:

- Tuning by experimenting the hyperparameters with ourselves
- RandomSearchCV
- Optuna

After testing many models, we decided to use **LightGBM** and **Random Forest** as our final models. We chose them because they gave us the best combination of accuracy, speed, and stability for this multiclass classification task.

Why We Chose LightGBM

- It gave high accuracy and handled imbalanced classes well.
- It trained very fast, even on a large dataset.
- It worked well with tuning and consistently improved when we adjusted hyperparameters.

Why We Chose Random Forest

- It's a very reliable and stable model with good general performance.
- It handled noise and overfitting better than single decision trees.
- It dealt well with imbalanced data.

After identifying LightGBM and Random Forest as top-performing models, we built a stacking ensemble to combine their strengths. In stacking, we used:

- **Base models**: LightGBM and Random Forest
- **Meta-model**: Logistic Regression (One-vs-Rest)

Stacking allowed us to blend the predictions of the individual models and make a final prediction using a meta-classifier. This technique helped capture different patterns each model learned, and improved generalization on the test data.

Model Evaluation:

After implementing each model, we tested it alone:

The LGBM alone accuracy was 0.9145

The Random forest alone accuracy was 0.9112

The final stacked model gave us the best test accuracy of 91.51% (0.915137). It was better than all individual models and other ensemble techniques like soft voting.

Conclusion:

In this project, we built a machine learning system to classify network traffic into different types, including both normal activity and various cyberattacks like DDoS, DoS, and others.

We tested many machine learning models and tried different techniques to improve performance. After experimenting with several combinations, we chose LightGBM and Random Forest as our best models. We then combined them using a stacking method to get even better results.

The stacking model gave us the highest accuracy and proved to be the most reliable. It successfully learned from complex patterns in the data and handled different types of attacks effectively. The final best accuracy was 91.51% (0.915137).

This model can be used as part of a larger network security system to help detect and respond to threats automatically.

Task division:

- Jana Ahmed: Feature engineering
- Ahmed Mohsen: Model implementation
- Ali Elrouby: Data investigation and exploration
- Mazen Ahmed: Model tuning
- All of us: report writing and investigation.