Cyber Security Attack Type Classification

GROUP 8

Filip Jakubowski:

filip.jakubowski@edu.dsti.institute

Mohamed Abdelkarim:

mohamed.abdelkarim@edu.dsti.institute

Younes Sogandi:

younes.sogandi@edu.dsti.institute

WANG Xiugang:

wang.xiugang@edu.dsti.institute

Ai Minh Nguyen:

ai-minh.nguyen@edu.dsti.institue

Supervisor: Hanna Abi Akl, Msc

March, 2025

I. Introduction

In this project, we analyze a cybersecurity dataset containing 40,000 records and 25 different metrics related to cyber-attacks. The dataset is raw and unprocessed, meaning it contains inconsistencies, missing values, and noise, making data preprocessing a crucial step.

Our goal is to train a machine learning model to predict the type of cyber-attack based on the given features. To achieve this, we will follow a structured machine learning pipeline, including Exploratory Data Analysis, Feature Engineering, Model training, and evaluation. The final model will be deployed as a web application, where users can input data through an interface and receive predictions on attack types.

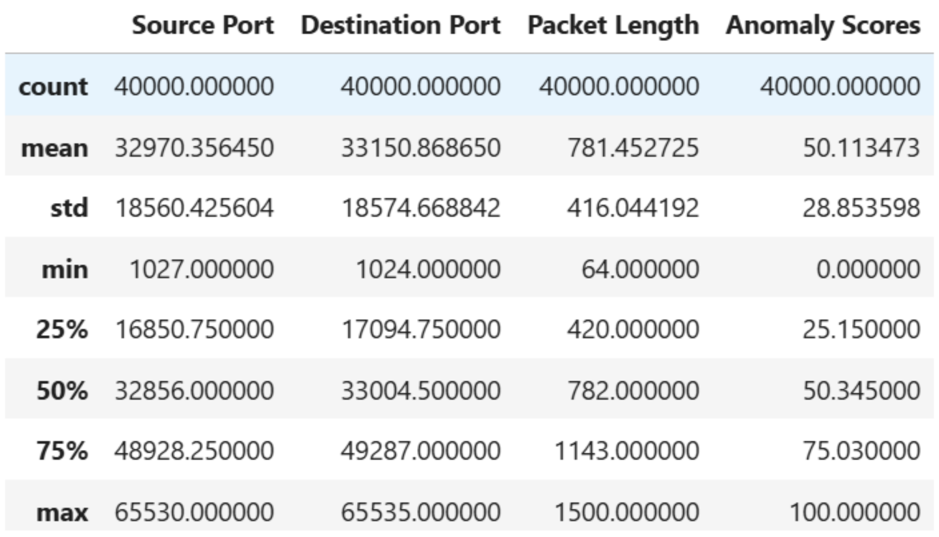
From our initial analysis, we observed that since most feature correlations are weak, successful attack detection will require careful feature engineering and model optimization rather than relying on simple linear relationships.

Through this project, we aim to develop a data-driven cybersecurity tool that enhances attack detection and classification, helping improve digital security defenses.

II. Methodology

1. Exploratory Data Analysis (EDA)

- Using df.describe()



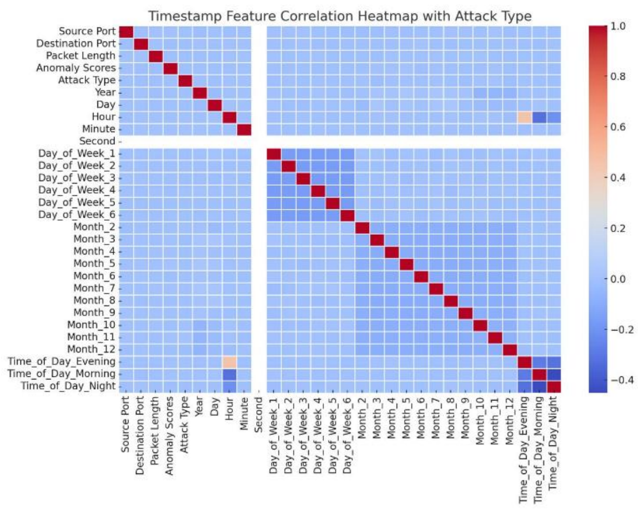
	Source Port	Destination Port	Packet Length	Anomaly Scores
count	40000.000000	40000.000000	40000.000000	40000.000000
mean	32970.356450	33150.868650	781.452725	50.113473
std	18560.425604	18574.668842	416.044192	28.853598
min	1027.000000	1024.000000	64.000000	0.000000
25%	16850.750000	17094.750000	420.000000	25.150000
50%	32856.000000	33004.500000	782.000000	50.345000
75%	48928.250000	49287.000000	1143.000000	75.030000
max	65530.000000	65535.000000	1500.000000	100.000000

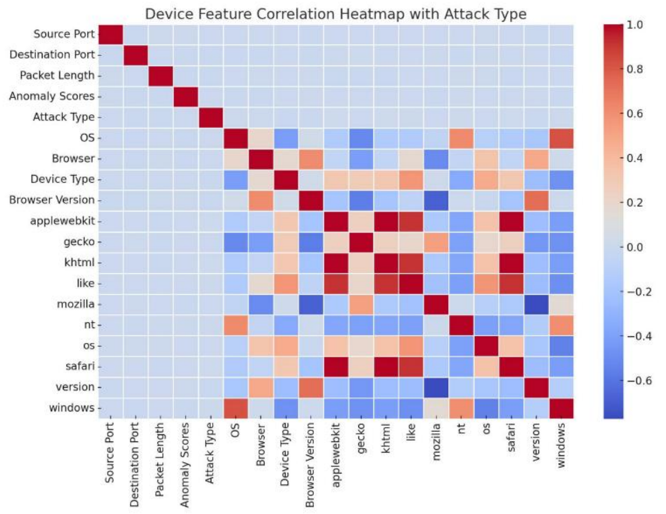
- Key observations:
  - Wide port ranges used: Both Source Port and Destination Port span nearly the entire valid port range, suggesting highly diverse or randomized traffic.
  - Packet length has broad variation: Sizes range from 64 to 1500 bytes, covering the full Ethernet frame size range. It is a sign of mixed traffic types, possibly from different protocols or behaviors.
  - Anomaly scores are evenly spread (0–100)

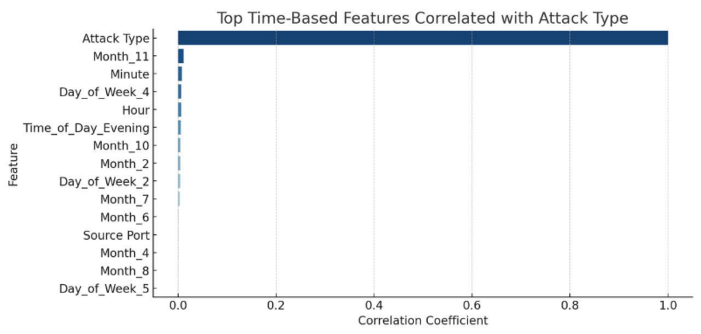
- Missing values in the following columns: Malware indicators, Alerts/Warnings, Proxy Information, Firewall Logs, and IDS/IPS Alerts
- Columns to remove because of irrelevance: User Information, Geo-location Data, Payload data

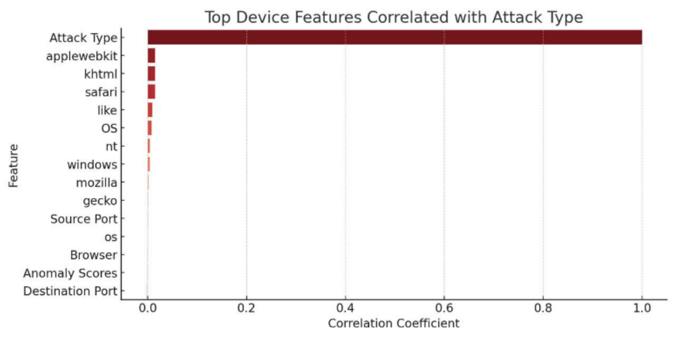
- The correlation matrix:

Timestamp and Device features in correlation heatmap with Attack Type









Based on these graphs, we have drawn these observations:

- Most features, whether timestamp-based (Hour, Month, Day_of_Week) or device/browser-related (OS, applewebkit, safari, etc.), show **very weak correlation with attack type**. No single feature stands out as a strong indicator.

This makes it hard to isolate threats using simple statistical or rule-based techniques.

- Many features, especially categorical ones like browser identifiers or operating system details, create a large number of dimensions—many of which are **redundant or weakly informative**. Sparse and noisy representations reduce clarity.

This increases complexity and can overwhelm learning models if not handled properly.

Conclusion: Raw data lacks strong predictive signal. Simple time or browser details alone do not distinguish attacks well.

2. Feature Engineering and Selection

- Fill missing values in the following columns: Malware indicators, Alerts/Warnings, Proxy Information, Firewall Logs, and IDS/IPS Alerts
- Convert categorical values into binary ones for Malware indicators, Alerts/Warnings, Firewall Logs, IDS/IPS Alerts, etc.
- Basic encoding for Protocol, Traffic type, Action taken, etc. using one-hot encoding.

- Device and network features:
  - Extract device info (OS, Browser, Device Type) from user-agent strings.
  - Extract IP Octets from source/destination Ips
  - Encode Proxy_Used as binary.
- Time-based features:
  - Extract from timestamp: year, month, day, hour, minute, second, day of week.
  - Categorize time into day periods (Morning, Afternoon, etc.) and encode.
- Port and size features:
  - Bin source and destination ports into known ranges (Well-Known, Registered, Dynamic) and one-hot encode.
  - Log-transform packet length to reduce skew.
- Multiply octets between source and destination IPs (e.g., Src_Octet1 * Dst_Octet1) to create network interaction features.

- Drop now-unnecessary columns: Source IP Address, Destination IP Address, and Proxy Information.
- Map attack types to integers
- Convert booleans to integers, standardize numerical features
- PCA and Feature Selection:
  - Apply PCA to reduce dimensionality (n_components=0.95).
  - Store top components and their variance contributions.
  - Train a RandomForestClassifier on principal components.
  - Extract top 20 most important PCs and save the final dataset.

3. Model training and evaluation

We propose 3 models:

1. Random Forest Classifier

Step 1: Data Preparation

- The preprocessed dataset is loaded.
- Features (X) and target labels (y) are separated.
- The data is split into training and testing sets using stratified sampling to preserve class distribution.

Step 2: Hyperparameter Tuning

- A parameter grid is defined with combinations of tree depth, number of trees, and split criteria.
- Grid Search with 5-fold cross-validation is used to find the best combination of parameters.

Step 3: Training the Best Model

- The optimal Random Forest model is trained on the full training set.

Step 4: Evaluation

- The model is evaluated on the test set using accuracy and a full classification report (precision, recall, F1-score).
- The model and report are saved for future use.

2. XGBoost Classifier

Step 1: Hyperparameter Tuning

- A new parameter grid is defined for gradient boosting parameters such as learning rate, tree depth, and subsampling rates.
- Grid Search with 5-fold cross-validation is used to identify the best XGBoost model configuration.

Step 2: Model Training and Evaluation

- The best XGBoost model is trained on the training set.
- Performance is measured on the test set using accuracy and a detailed classification report.
- Both the trained model and report are saved.

3. Feedforward Neural Network (PyTorch)

Step 1: Tensor Conversion and Data Loader Creation

- The training and test sets are converted into PyTorch tensors.
- A data loader is created to handle mini-batch training.

Step 2: Model Definition

- A neural network is defined with two hidden layers, ReLU activations, and a softmax output for multi-class classification.

Step 3: Training Loop

- The model is trained over multiple epochs using the Adam optimizer and cross-entropy loss.
- Loss is printed periodically for monitoring.

Step 4: Evaluation

- The trained model is evaluated on the test set by predicting class probabilities and selecting the most likely class.
- Accuracy and a classification report are generated and saved.
- The trained model weights are saved for reuse.

Results

Given the nature of the dataset, where earlier analysis showed very weak correlations between individual features and the attack type, all three models are our best models. Despite feature engineering efforts, the signal remains low, and no clear pattern strongly predicts the target variable. The three models we proposed achieved similar accuracy rates around 33%, which is near random guessing for a 3-class classification problem. XGBoost performed slightly better with 33.72%, followed by the Neural Network at 33.38% and Random Forest at 33.02%.

Deployment

The web application provides a simple and user-friendly interface for performing cyberattack prediction. Users can upload a CSV file containing preprocessed data and select one of the three available models—Random Forest, XGBoost, or Neural Network—for prediction. Once a model is chosen, clicking the "Upload and Predict" button runs the selected model on the uploaded data and returns the predicted attack types. This streamlined workflow allows users to quickly evaluate different models and analyze outcomes.

Conclusion

To enhance performance, future work could focus on incorporating richer contextual data, such as session-level behavior, network flow patterns, or external threat intelligence. Improving data quality, balancing class distributions, and exploring advanced ensemble methods may also provide better predictive power. Additionally, integrating domain knowledge into feature design could help uncover hidden patterns that machine learning models alone may not detect.

Github link: https://github.com/m-ihab/cybersecurity-prediction-app.git

The files in our Github include:

Data Engineering.ipynb	Official file about our EDA and feature engineering processes
Modeling.ipynb	Official file about our modeling processes
Detailed process – NO RESULT.ipynb	Extra file about our whole processes in detailed
Video – App's performance	A short video about the web application

Group work repartition

Full Name	**Tasks**
Filip Jakubowski	EDA, Feature Engineering, Modeling, Web app
Mohamed Abdelkarim	EDA, Feature Engineering, Github, Web app
Younes Sogandi	EDA, Feature Engineering, Modeling
Wang Xiugang	EDA, Feature Engineering, Report
Ai Minh Nguyen	EDA, Feature Engineering, Report