Cyber Security Attack Type Detection

Prepared by:
Abir Lakhal (DE)
Katy Mayoro Fall (DE)
Amina Hocine (DA)
Amina Gulzar (DA)
Nathalie Colard (DS)
Sehomi ESSENOUWA (DA)

Instructor: Hanna Abi Akl

Python Machine Learning Labs

Table of contents

Introduction

In an era where cybersecurity threats are increasingly complex and significant, it is essential to develop robust mechanisms to detect and mitigate cyberattacks. This project aims to build a machine learning model that can predict the type of cyberattack based on network and security parameters. The provided dataset contains 40,000 records and 25 attributes including network traffic details, attack patterns, anomaly scores, firewall logs, or IDS/IPS alerts. The raw nature of the data requires extensive preprocessing (EDA), feature selection, and model optimization.

The main objective is to build an end-to-end ML pipeline that processes the dataset, extracts meaningful insights, and trains an optimal model for attack classification. Additionally, the final model is deployed as a web application, providing an interface where users can input data and receive attack type predictions in real-time. The implementation follows standard ML practices including exploratory data analysis (EDA), feature engineering, model selection and evaluation, ensuring a rigorous approach to cybersecurity threat detection.

1 Methodology

To achieve the project objectives, we followed a structured machine learning pipeline that involved several key steps, starting with Exploratory Data Analysis (EDA) to better understand the dataset and identify important features for modeling.

1.1 Exploratory Data Analysis

A thorough analysis of the dataset was conducted to identify patterns and potential predictors for attack classification.

The target variable is “Attack Type”, categorized into:

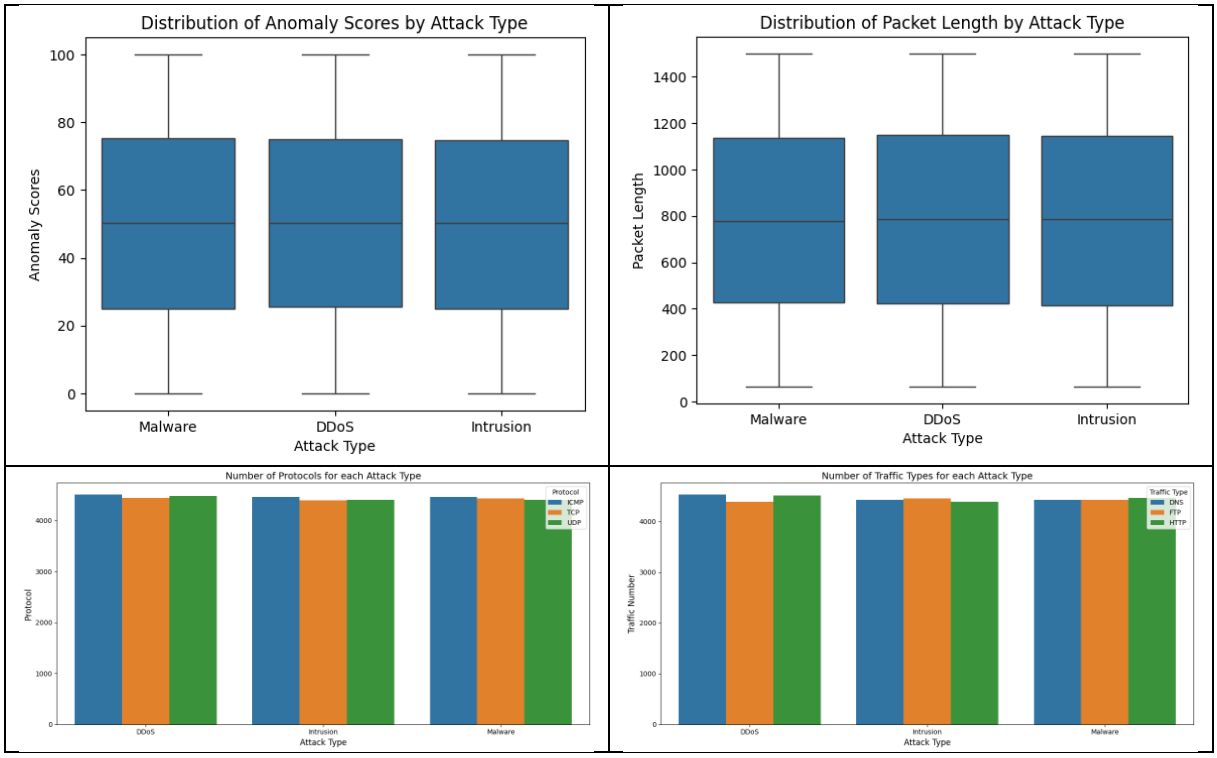
```
Attack Type
DDoS         13428
Malware      13307
Intrusion    13265
Name: count, dtype: int64
```

From the exploratory data analysis (EDA), several key points can be deduced regarding the dataset and its ability to predict types of cyberattacks:

➢ Lack of Strong Predictive Features:

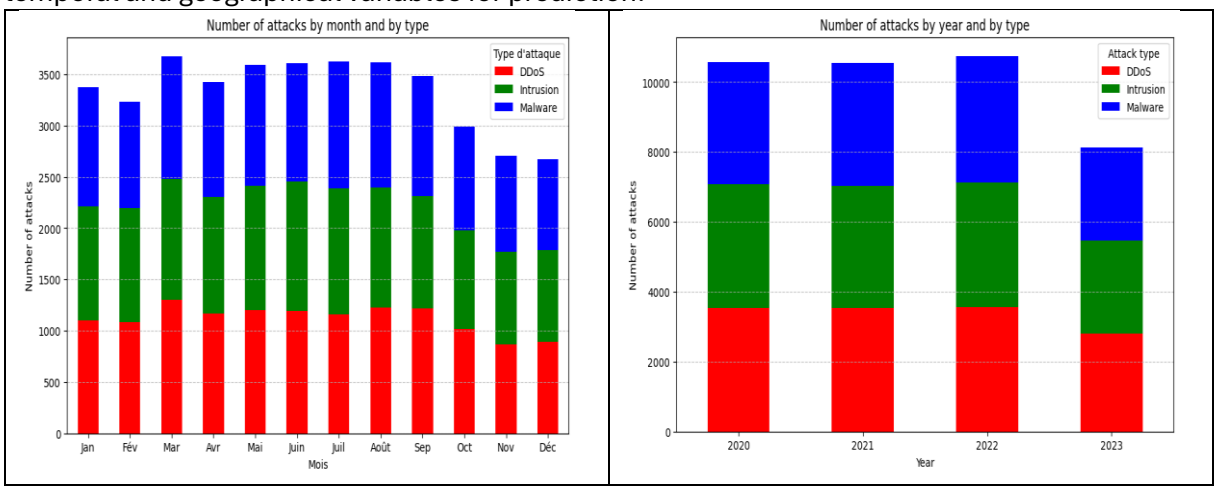
No feature clearly stands out as a strong predictor for classifying attack types.

The distributions of numerical features (such as anomaly scores, packet length) and categorical features (such as protocols, traffic types) are very similar across the three attack types, limiting their usefulness for modeling.



➢ Absence of Temporal Trends

Attacks appear to occur uniformly over time (no seasonal or annual trends) and the geolocation data does not show significant concentration in specific regions. This reduces the relevance of temporal and geographical variables for prediction.



➢ Weak Correlations Between Features

Correlation analysis and mutual information (MI) scores show that relationships between features are weak. Even features with relatively high MI scores (such as source and destination ports) do not show discriminative distributions across attack types.

The data exploration revealed that the relationships between variables are complex and interconnected, but no single feature stands out as a key predictive factor. This highlights the importance of feature engineering and and deeper analysis to refine our approach and enhance model performance. The next steps would involve experimenting with different algorithms and adding more domain-specific knowledge to bring clarity to this complex, multifaceted dataset.

1.2 Feature Engineering and Selection

Feature engineering is a critical step in preparing the data for machine learning models. It involves handling missing values, removing irrelevant features, parsing and transforming data columns, and encoding categorical variables to ensure that the models can extract meaningful patterns from the data.

➢ Handling Missing Values

In the dataset, several columns contain missing values. We replace these missing values with appropriate substitutes based on the context of the column. For instance:

- Firewall Logs are filled with "No Log"
- Proxy Information is replaced with "No Proxy"
- Malware Indicators are filled with "No IoC Detected"
- Alerts/Warnings are replaced with "No Alert Triggered"
- IDS/IPS Alerts are filled with "No Alert Data"

This ensures that the model does not encounter NaN values, which could hinder its performance.

➢ Removal of Irrelevant Columns

Irrelevant columns, such as Source IP Address, Destination IP Address, Source Port, and others, are dropped from the dataset to avoid unnecessary noise in the model. These features do not provide significant information for the task at hand.

➢ Parsing the "Timestamp" Column

The Timestamp column is parsed to extract the day, month, and year of each observation. This provides more granular information about the event's occurrence, which might be relevant for the analysis. The original Timestamp column is then removed.

➢ Parsing the "Device Information" Column

The Device Information column is parsed to extract details about the operating system, device, device brand, device model, and browser used. This helps provide additional context about the device involved in the event. After parsing, the Device Information column is dropped.

➢ Parsing the "Geo-location Data" Column

The Geo-location Data column is parsed to extract the state information. The original column is then dropped as it is no longer needed)

➢ Encoding Categorical Variables

To make the data suitable for machine learning models, categorical columns such as Protocol, Packet Type, and Malware Indicators are one-hot encoded. This transforms the categorical data into numerical format, which allows the model to learn from these features.

➢ Splitting the Data

After encoding the data, the dataset is split into features (X) and target (y). The target variable is the Attack Type, which will be predicted by the model. The features include all the one-hot encoded columns.

➢ Feature Selection using Recursive Feature Elimination (RFE)

Recursive Feature Elimination (RFE) is used to select the most relevant features for the model. RFE eliminates the least important features iteratively based on the performance of a machine learning model. We apply RFE using three different models: DecisionTreeClassifier, RandomForestClassifier, and LogisticRegression.

DecisionTreeClassifier: RFE selects the top 20 features.

RandomForestClassifier: RFE selects the top 20 features.

LogisticRegression: RFE selects the top 20 features.

After applying RFE, the models are trained using only the selected features. The performance of the models is evaluated on the test data, and a classification report is generated.

The features selected by RFE allow the model to focus on the most relevant data, improving performance while reducing computational complexity.

1.3 Model Selection, Comparison and Evaluation

In this project, we tested three machine learning models to predict cybersecurity attack types: **Decision Tree**, **Random Forest**, and **Logistic Regression**. These models were chosen for their ability to handle classification tasks effectively and provide different insights into the data.

We implemented the following steps to evaluate the models:

- ### Model Training:
  We trained each model using the training dataset (`X_train` and `y_train`).

- ## Model Evaluation:

  After training, we used the test set (`X_test`) to make predictions. We then assessed the performance of each model using the following metrics:

  - **Accuracy:** Measures the proportion of correctly predicted instances.
  - **Classification Report:** Provides precision, recall, and F1-score for each class.
  - **Confusion Matrix:** Shows the true positives, true negatives, false positives, and false negatives, helping visualize how well the model performs across different classes.

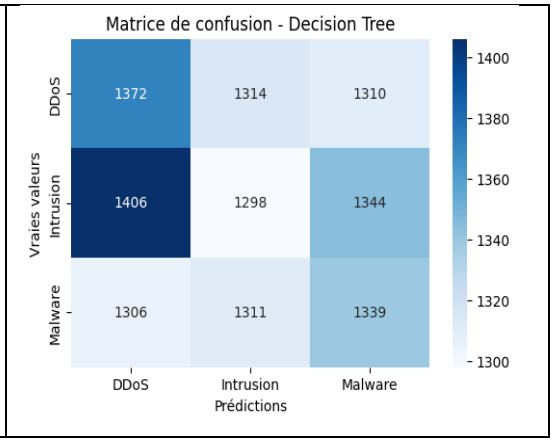
The models were evaluated using the `train_and_evaluate` function, which handles the training, predictions, and metric generation for each model. The confusion matrices were visualized with a heatmap for easy interpretation of performance.

We then evaluated the following models:

- **Decision Tree**

Classification Report:

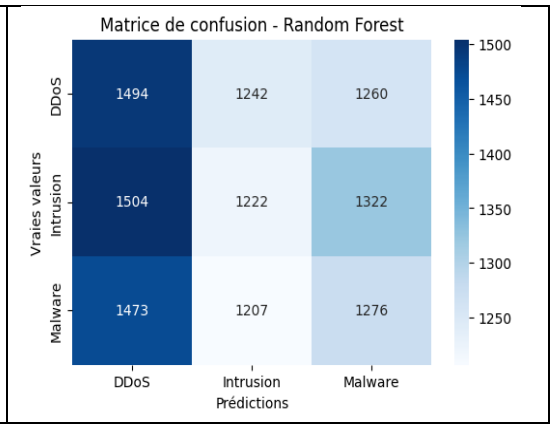
	precision	recall	f1-score	support
DDoS	0.34	0.34	0.34	3996
Intrusion	0.33	0.32	0.33	4048
Malware	0.34	0.34	0.34	3956
accuracy			0.33	12000
macro avg	0.33	0.33	0.33	12000
weighted avg	0.33	0.33	0.33	12000



- **Random Forest**

Classification Report:

	precision	recall	f1-score	support
DDoS	0.33	0.37	0.35	3996
Intrusion	0.33	0.30	0.32	4048
Malware	0.33	0.32	0.33	3956
accuracy			0.33	12000
macro avg	0.33	0.33	0.33	12000
weighted avg	0.33	0.33	0.33	12000

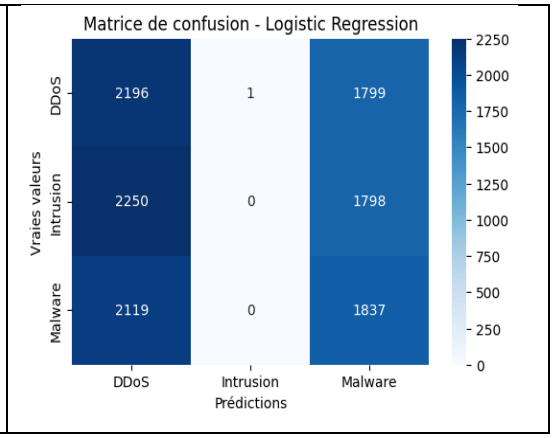


- **Logistic Regression**

```
Classification Report:
              precision    recall  f1-score   support

       DDoS       0.33      0.55      0.42      3996
   Intrusion      0.00      0.00      0.00      4048
     Malware      0.34      0.46      0.39      3956

    accuracy                           0.34     12000
   macro avg       0.22      0.34      0.27     12000
weighted avg       0.22      0.34      0.27     12000
```



2 Results

After evaluating the three models, we found that the Decision Tree model provided the best performance for our use case. Although the Random Forest model showed slightly better accuracy, the Decision Tree was preferred due to its simplicity, interpretability, and consistent performance across different metrics. The Logistic Regression model, while effective in many scenarios, did not perform as well in this particular case.

Thus, we decided to keep the Decision Tree model as the final choice for the application, as it offers both high performance and ease of interpretation without the need for additional techniques like Recursive Feature Elimination (RFE). The model has been saved as a `.pkl` file and is ready for deployment in the web application, ensuring smooth and reliable predictions for users.

3 Deployment

This web application, built using Python and Streamlit, allows users to predict the type of cybersecurity attack effortlessly. The app deploys a machine learning model that makes predictions based on user input or uploaded data. It offers two prediction methods:

- **Manual Prediction:**
  Users can input details through the interface and click “Predict” to receive an instant attack type prediction.
- **CSV File Prediction:**
  Users can upload a CSV file, ensuring it follows the dataset format (excluding the "Attack Type" column). The app processes each line and displays the predicted attack type.

Streamlit provides a simple yet powerful interface, allowing for easy interaction and making cybersecurity predictions accessible to everyone.

Conclusion

This project was a significant challenge due to the raw and imperfect nature of the dataset, which contained 25 features and 40,000 rows. The team worked collaboratively across different disciplines to preprocess the data, perform feature engineering, and train a machine learning model. Despite the complexities, we successfully built an end-to-end pipeline that predicts cyber-attack types and deployed a user-friendly web application.

The project provided valuable insights into data handling, model selection, and deployment. Moving forward, there's potential to improve by expanding the dataset, optimizing the model, and refining the application's interface. Overall, this experience enhanced our skills in teamwork, problem-solving, and machine learning.