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

GitHub link: https://github.com/aminahocine/CyberSecurity_AttackType_Detection.git

Python Machine Learning Labs

Contents

Cyber Security Attack Type Detection

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 (DDoS, Malware or Intrusion) 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.

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.

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
```

EDA aims to understand the dataset and find correlations between features to help predict the type of attack:

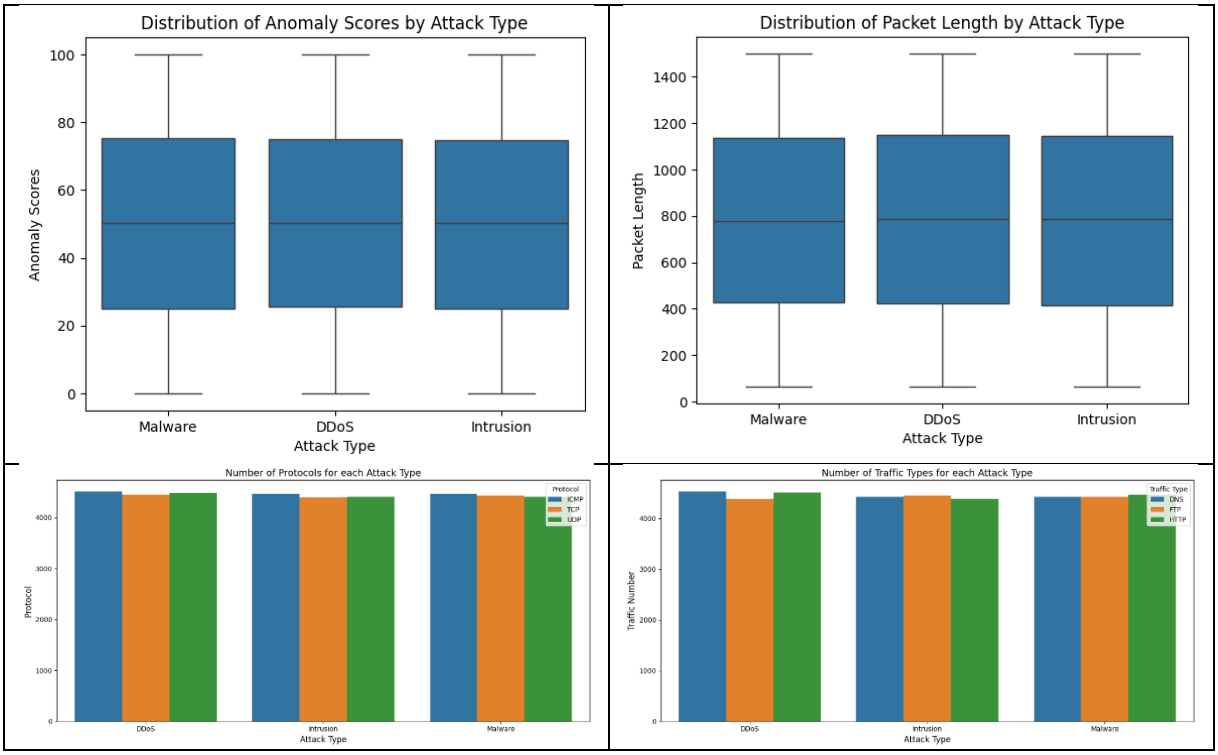
Lack of Strong Predictive Features:

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

Some features do not provide any valuable insights for predicting the type of attack, as they have as many unique values as there are observations. (For example: Source IP Address, Destination IP Address, Payload Data, User Information)

We analyzed both categorical and numerical columns:

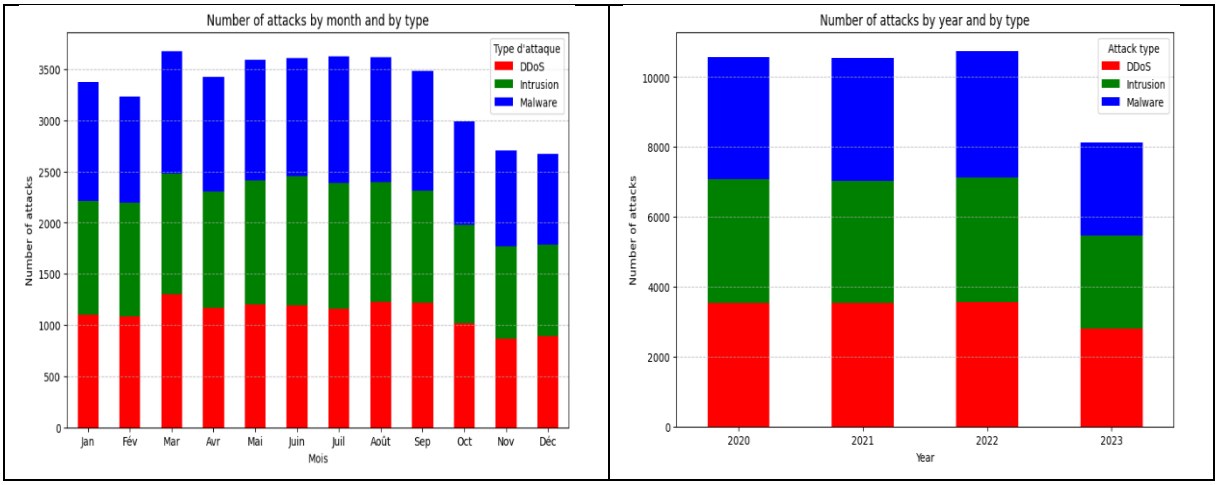
The distributions of numerical features (such as anomaly scores, packet length) and categorical features (such as protocols, traffic types) are very similar across the three attack types, limiting their usefulness for modeling.



So, for the categorical columns Protocol, Packet Type, Traffic Type, Attack Signature, Action Taken, Severity Level, Network Segment, Log Source, Malware Indicators, Alerts/Warnings,Firewall Logs and IDS/IPS Alerts the values are evenly distributed across the three types of attacks. There do not appear to be any standout values that could help predict one type of attack over another.

Lack of temporal and geographical trends

We analyzed the Timestamp column to see if a type of attack is more present according to a certain temporality. But here too, we found uniform results



For the geographical aspect, we analyzed the Geo-location Data column. We noticed that there is no particular predominant state for each type of attack. Moreover, in the top 10 of each attack, certain states recur in all 3 attacks (Maharashtra, Bihar, Manipur ..)

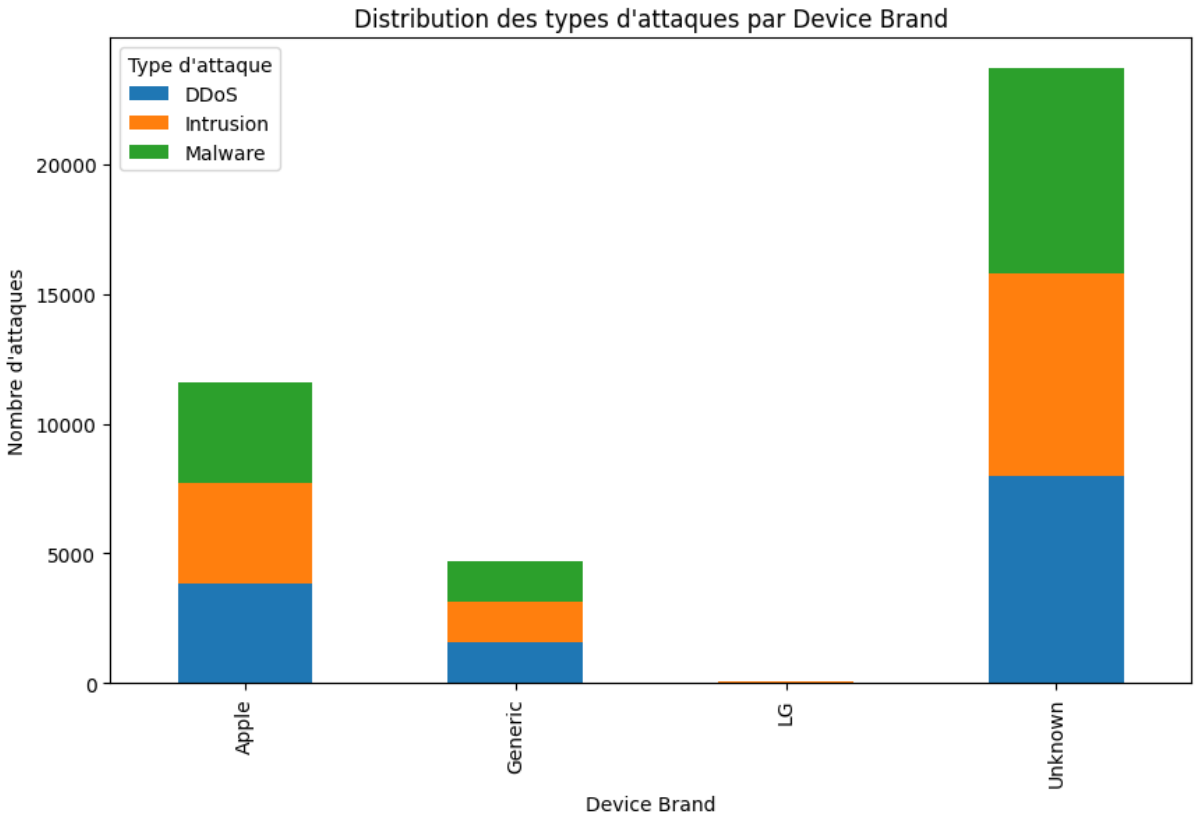
```
Top 10 localisations States pour Malware:
State
Gujarat          508
Sikkim           498
Uttar Pradesh    498
West Bengal      498
Maharashtra      495
Bihar            493
Haryana          491
Assam            488
Punjab           485
Mizoram          483
Name: count, dtype: int64

Top 10 localisations States pour DDoS:
State
Manipur             522
Karnataka           520
Arunachal Pradesh   502
Maharashtra         499
Rajasthan           497
Bihar               497
Meghalaya           495
Himachal Pradesh    493
Odisha              491
Uttar Pradesh       490
Name: count, dtype: int64
```

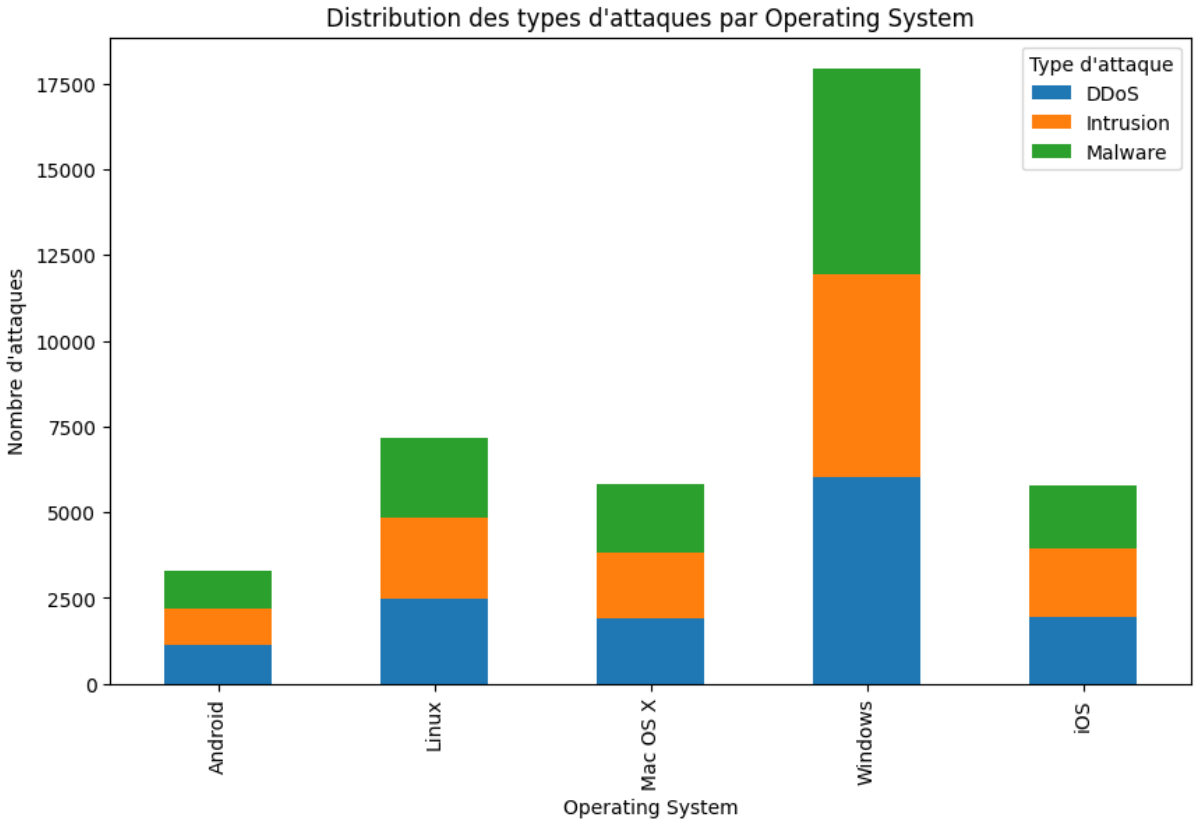
```
Top 10 localisations States pour Intrusion:
State
Jharkhand            526
Uttarakhand          520
Manipur              500
Arunachal Pradesh    499
Nagaland             497
Uttar Pradesh        497
Gujarat              495
Rajasthan            486
Maharashtra          480
Mizoram              480
Name: count, dtype: int64
```

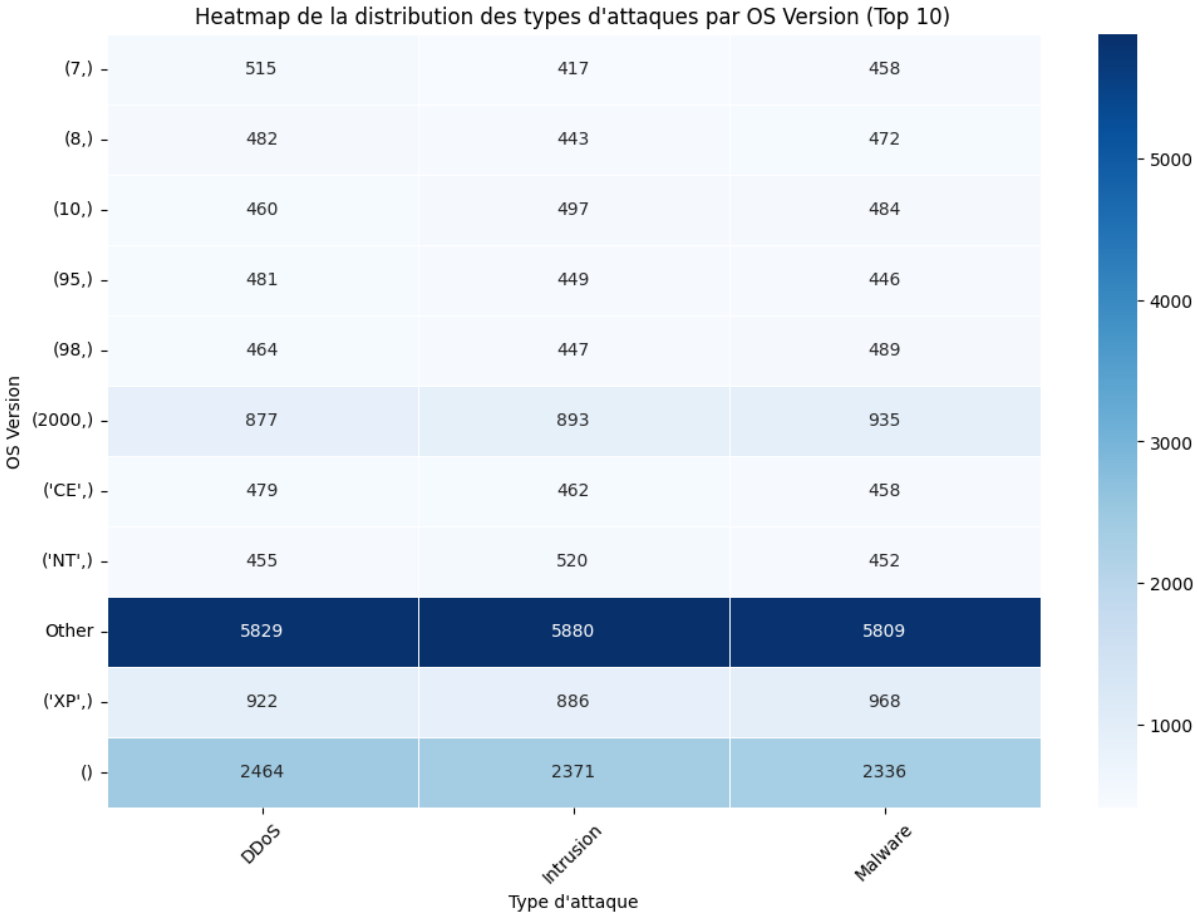
This reduces the relevance of temporal and geographical variables for prediction.

Uniformity in device information column

We analyzed the Device Information column, for this we use the user_agents library to parse the column information.



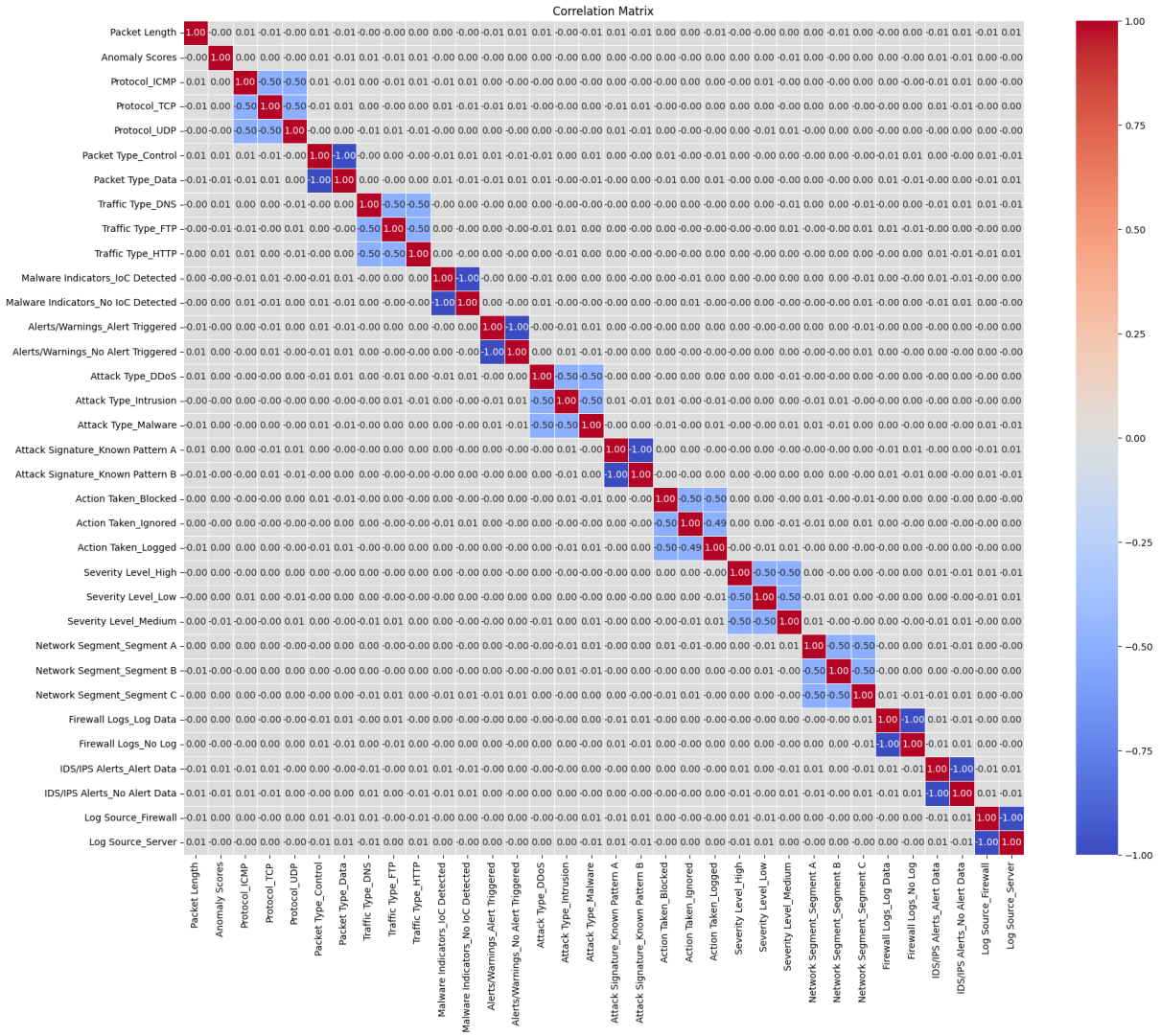




We found uniformity in the distribution of attacks across several device information parameters.

Correlation Matrix & Mutual information Algorithm: Weak Correlations Between Features

We plotted the correlation matrix:



We tried the MI algorithm and according to the results, we should have kept the columns: Source Port (0.85), Destination Port (0.85), Anomaly Scores (0.30). But we saw in the previous analyses that each of these columns had a uniform distribution and are not relevant for attack type prediction

Thus, 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 uniform, but no single feature stands out as a key predictive factor. This highlights the importance of feature engineering and deeper analysis to refine our approach and enhance model performance. The next steps is feature engineering and selection.

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, 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. In our case, we noticed that the RFE does not allow us to improve the performance of our models (precision of ~33% for each attack) compared to the result we would have without feature selection.

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:

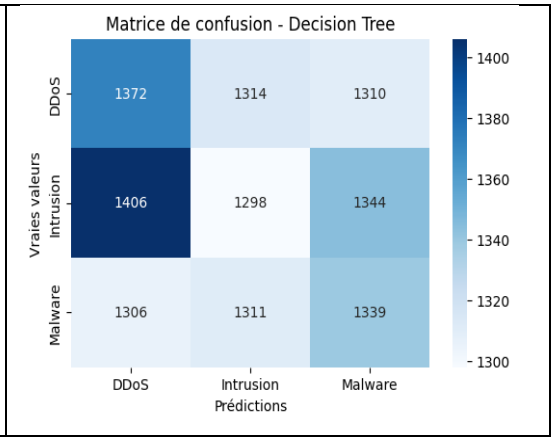
  - **Accuracy:** Measures the proportion of correctly predicted instances.
  - **Classification Report:** Provides precision, recall, and F1-score for each class.
  - **Confusion Matrix:** Shows the true positives, true negatives, false positives, and false negatives, helping visualize how well the model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

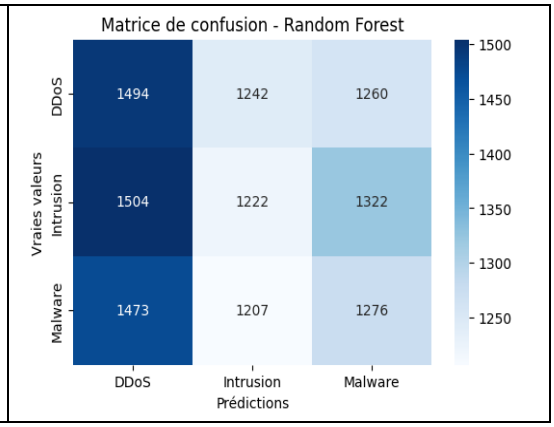
- **Decision Tree**

```
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
```



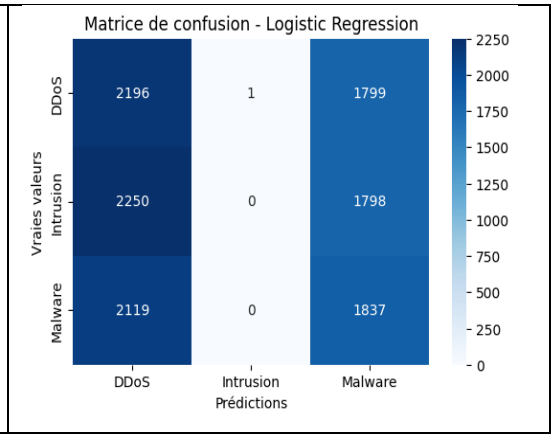
- **Random Forest**

```
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
```



Results

After evaluating the three models, we found that the Decision Tree model without RFE provided the best performance for our use case. But the model's performance still remains very poor. This is explained by the fact that during the EDA, we noticed a uniform dataset without particular trends and correlations between features allowing to predict the type of attack. This is reflected in the model's performance since it randomly predicts each time one attack out of the 3 existing ones (precision of ~33%)..

Thus, we decided to keep the Decision Tree model without RFE as the final choice for the application, since it offers better performance (by a few %) compared to the addition of the feature selection technique that the Recursive Feature Elimination algorithm offers us. 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. All steps discussed so far in this report are in the Jupyter Notebook "Projet_Cybersecu_ML_final_version.ipynb" in the "project" folder in Github.

Deployment

This web application, built using Python and Flask, 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. To predict, go to the "Login" page and enter "ANASKA" as the username and "cyberA24" as the password. 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. Examples of .csv files are available in github (in projet folder) as “testpredict.csv” or “testpredict-several-lines.csv"

The result will be displayed at the bottom of the page:

A short demo video of the application is available in the github in the "project" folder under the name "Cybervault Demo Video.mp4"

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. Due to the uniformity of the dataset, our model predicts a random result, which means that we do not exceed 34% accuracy. We could improve this by adding more information, in particular adding other relevant features that could help predict the type of attack for example. 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.