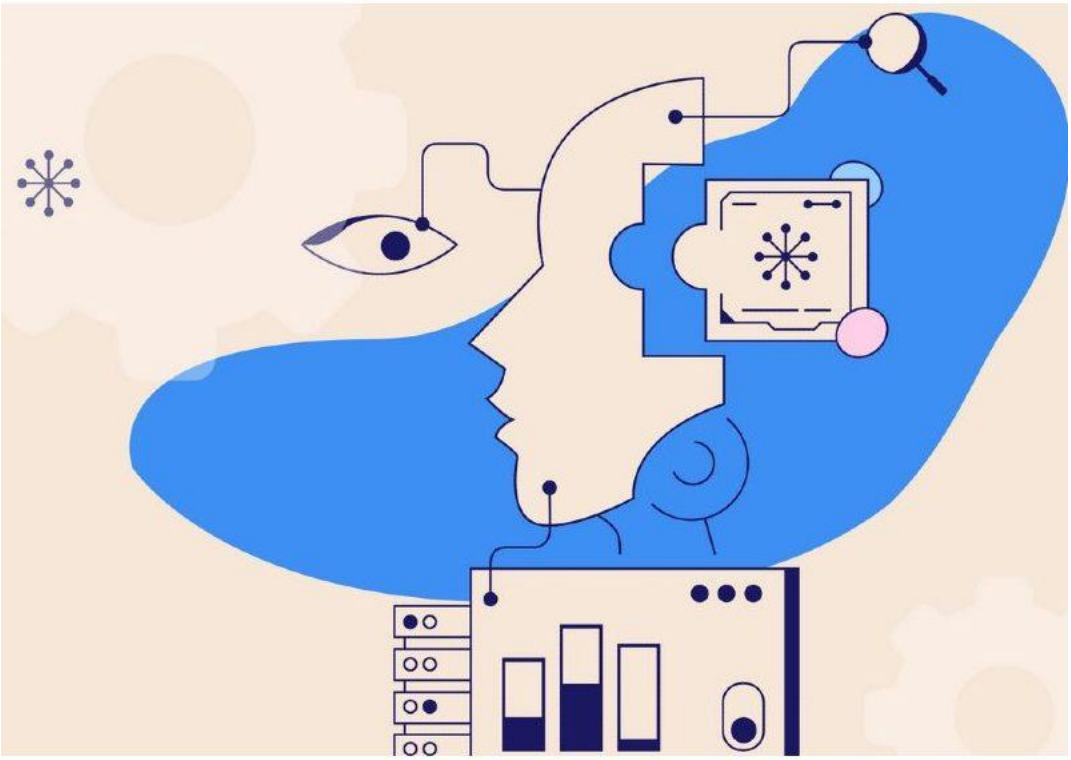
MACHINE LEARNING WITH PYTHON LABS

Group #7 - Project 1 : Cybersecurity Attacks



Data ScienceTech Institute

Anthony Baudchon
Marie-Caroline Bertheau
Gaia Bianciotto
Yasmine El-Khdar
Giti Shekari
Kokou Aubain Viglo

Table des matières

Introduction

In a world where cyberattacks are increasing and becoming more sophisticated, cybersecurity is a strategic challenge for businesses and organizations. Early detection of attacks is essential to limit potential damage and ensure the protection of information systems. However, the diversity and complexity of threats make this task difficult and require advanced approaches based on artificial intelligence and machine learning.

This project is part of this dynamic by proposing a solution to automatically identify the type of attack from a set of raw data. The main objective is to develop a complete machine learning pipeline to accurately predict the type of cybersecurity attack based on a set of provided measures. The dataset provided includes 40,000 entries and 25 attributes, ranging from network information (IP addresses, ports, protocols) to security indicators (attack signatures, anomaly scores, firewall logs).

To do this, several steps will be carried out: exploratory data analysis to understand the patterns and relationships within the dataset, feature engineering and selection to identify and transform the predictive variables, thereby improving the performance of the model, training and evaluation of various machine learning models, selection of the best performing model based on appropriate evaluation measures. Finally, a web application will be deployed to facilitate the prediction of attack types, allowing users to enter data and receive real-time predictions.
GitHub Link:
https://github.com/AnthonyBaudchon/Cybersecurity_Attacks/tree/main

By integrating these different dimensions, our work aims to demonstrate the relevance of machine learning techniques in the field of cybersecurity and to provide a practical application that can help in the rapid identification of cyberattacks.

1. Methodology

I. Data Description

Our dataset consists of 40000 rows and 25 columns, capturing Cyber Attacks. The features can be classified as:
Network-related features: Source IP Address, Destination IP Address, Source Port, Destination Port and Protocol
Traffic characteristics: Packet Length, Packet Type and Traffic Type
Security indicators: Malware Indicators, Anomaly Scores, Alerts/Warnings, Attack Signature and Severity Level
System logs & metadata: Firewall Logs, IDS/IPS Alerts, Proxy Information and Geo-location Data.

II. Exploratory Data Analysis (EDA)

As we have only two numerical features, a correlation matrix or statistical summary may not be the most insightful approach. However, these features indeed indicate that the dataset includes both small and large packets with most packets being of moderate size. Likewise, the anomaly detection system assigns scores ranging from 0 (no anomaly) to 100 (maximum anomaly), and on average, packets show a moderate level of anomaly (50).

We can notice that most features are fully populated except for: **Malware Indicators**, **Alerts/Warnings**, **Proxy Information**, **Firewall Logs**, and **IDS/IPS Alerts**, each with around **50% missing data**.
Our dataset includes both high and low cardinality features. **Source IP**, **Destination IP**, **Source Port**, **Destination Port**, **User Information**, and **Device Information** show high variation and may require extra processing. On the other hand, **Protocol**, **Traffic Type**, **Attack Type**, and **Severity Level** have only three unique values and are more suitable for encoding. **Malware Indicators**, **Firewall Logs**, and **IDS/IPS Alerts**, with just one unique value, may not contribute to model learning.
Furthermore, the dataset is well-balanced, with each attack type **DDoS, malware, and intrusion** having approximately **13,400 rows**. which ensures that the model is not biased toward any specific class, allowing for fair training and evaluation.

The dataset reveals a balanced distribution of attack types across various attributes, including **protocols, traffic types, and network segments**, with no strong correlation to any specific category. **Geographically**, some cities and states experience slightly higher attack frequencies, highlighting potential cybersecurity hotspots. **Temporally**,

attack distributions remain stable across years, though a slight decline is observed in 2023. Monthly and hourly trends show minor fluctuations but no significant attack patterns. Lastly, **user behavior analysis** indicates that most users appear uniformly across attack types, though some outliers may warrant further investigation.

III. Feature Engineering (FE)

As we noticed before, the columns with missing values each contained only one unique value, making it nearly impossible to predict anything other than the absence of that value. As a result, negating it made sense and emerged as the most logical choice. Therefore, **IDS/IPS Alerts** were filled with **"No Alert"**, while **Malware Indicators** were set to **"No IoC Detected"**. Similarly, missing values in **Firewall Logs** were replaced with **"No Log Data"**, **Alerts/Warnings** with **"No Alert Triggered"**, and **Proxy Information** with **"No Proxy"**. These replacements ensure data completeness and standardization, making the dataset more reliable for analysis and modeling.

Also, the dataset was refined through subnet analysis by extracting the **first two octets** of **Source IP**, **Destination IP** and **Proxy** to identify subnet patterns. Subnets with high occurrences were flagged as suspicious. After extraction and analysis, the **Source IP**, **Destination IP** and **Proxy** columns were removed.

The **'Geo-location Data'** column was split into two new columns: **'City'** and **'State'**. To enhance the analysis, the geopy library was used to obtain geolocation coordinates for each city. However, after visualizing the data on a map, geographical insights were not significant for our analysis. As a result, only the **City** and **State** columns were retained, and the original **'Geo-location Data'** column was discarded.

The **'Timestamp'** column was converted into a **datetime format**. From this, new features such as **year**, **month**, **day**, and **hour** were extracted. The **hours** were then categorized into **Morning**, **Afternoon**, **Evening**, and **Night**. Additionally, the **quarters** were mapped to seasons (Winter, Spring, Summer, Autumn). These transformations enhance the dataset's temporal understanding, providing better insights for analysis.

The **'Source Port'** and **'Destination Port'** columns were categorized into predefined ranges: **Well-Known Ports** (0-1023), **Registered Ports** (1024-49151), and **Dynamic/Private Ports** (49152-65535). Any out-of-range values were labeled as **Invalid Ports**. This categorization resulted in the creation of two new columns, Source Port Category and Destination Port Category, which allow for more targeted analysis based on port types, while the original port values were retained for reference.

The **'Packet Length'** feature was categorized based on size: **Small** (≤ 512 bytes), **Medium** (513 to 1024 bytes), **Large** (> 1024 bytes), and **Invalid** for non-numeric or invalid length values. This categorization resulted in the creation of a new column, **Packet Length Category** and the original packet length values were retained for reference.

The **'Anomaly Scores'** feature was categorized into three levels: Low Anomalous (scores below 30), Slightly Anomalous (scores between 30 and 49), and Highly Anomalous (scores 50 and above). This categorization resulted in the creation of a new

column, Anomaly Category and the original anomaly scores were retained for reference.
The **'Device Information'** column was processed using both the ua-parser library and custom regex patterns to extract key device and browser details such as browser name, operating system, device type, and rendering engine. The operating system values were further normalized to ensure consistency (e.g., grouping Android versions). After extracting and normalizing the information, the original 'Device Information' column was dropped and replaced.

IV. Models: Training and Validation

To predict the type of attack, we will employ several models Logistic Regression (a linear model for classification), Random Forest (an ensemble method that builds multiple decision trees), Decision Tree (a simple yet interpretable tree-based classifier), Support Vector Machine (SVM) (which is highly effective in high-dimensional spaces), and K-Nearest Neighbors (KNN) (an instance-based classifier). For each model, we will train it on the training data and evaluate its performance using test data.
The models will be evaluated based on key metrics such as accuracy and classification report, which include precision, recall, and F1-score.
Given the dataset's largely categorical features, we will also explore alternatives to clustering. While traditional K-Means struggles with categorical data, HDBSCAN emerged as a strong candidate due to its density-based approach and flexibility with appropriate distance metrics, potentially complementing our classification efforts.

2. Presentation of Results and Model Performance

I. Model Results

Summary of Tested Models and Their Functioning

In this project, several Machine Learning models were evaluated to classify cyberattacks. Below is an overview of each tested model, along with a simplified explanation of how it works.

- **Logistic Regression:**
    - **Description**

Logistic regression is a supervised classification algorithm used to predict a category (binary or multi-class) by modeling the probability that an observation belongs to a given class.

- **Functioning:**
  - Uses a sigmoid function to transform input values into probabilities.
  - Assigns a class label based on a threshold (generally 0.5 for binary classification).
  - Works well for linear problems but struggles with complex or non-linear data.

- **K-Nearest Neighbors (KNN - k nearest neighbors):**

  - **Description**

KNN is a proximity-based algorithm that classifies a new element based on the categories of the "k" closest points.

  - **Functioning:**
    - When a new sample arrives, the algorithm identifies the **k** closest points in the training set.
    - It assigns the majority class label among these neighbors.
    - The higher the value of **k**, the more robust the classification, but it may lose local precision.

- **HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise)**

  - **Description**

HDBSCAN is an unsupervised clustering algorithm used to identify natural groupings in data without requiring predefined labels.

  - **Functioning:**

    - Analyzes the local density of points to group similar samples.
    - Isolated points are considered "noise" and are not assigned to a cluster.
    - Very useful for detecting hidden patterns in complex data.

- **HDBSCAN & XGBoost (Combination of Clustering & Advanced Supervised Model)**
  - **Description:**

This approach combines **HDBSCAN** (to detect similar attack groups) with **XGBoost**, a powerful supervised model based on decision trees.

- **Functioning:**
  - HDBSCAN segments the data into representative clusters.
  - XGBoost uses these clusters to optimize its predictions.
  - XGBoost is a gradient boosting algorithm that iteratively trains multiple decision trees to minimize errors.

- **HDBSCAN & XGBoost & Optuna (Combination of Clustering & Advanced Supervised Model & Hyperparameters)**
  - **Description and functioning:**

This approach combines HDBSCAN for identifying similar attack groups with XGBoost, a powerful tree-based supervised model. Hyperparameters—such as learning rate, tree depth, subsampling, and regularization terms—were carefully optimized to enhance performance and prevent overfitting. Techniques like Grid Search, Random Search, and cross-validation were applied to find the best configurations, ensuring robust and accurate predictions.

II. Comparison of Model Performance

The evaluation of models was based on **accuracy**, which represents the percentage of correct predictions. Below are the obtained results:

Model	Accuracy (%)
Logistic Regression	34.49%
K-Nearest Neighbors (KNN)	33.51%
HDBSCAN	49.81%
HDBSCAN & XGBoost	69.83%
HDBSCAN & XGBoost & Optuna	**91%**

- **Performance Analysis :**

  - Logistic Regression and KNN obtained low results **(~34%),** proving they are not suited for the complexity of the dataset.
  - HDBSCAN alone slightly improved accuracy **(49.81%)**, but its purely unsupervised approach limited its precision.
  - **HDBSCAN & XGBoost offered the best performance (69.83%)**, demonstrating that combining clustering with a supervised model enhances attack classification.

- **HDBSCAN & XGBoost & Optuna resulted in a model with 91%** accuracy, indicating good overall performance. However, this metric should be complemented with precision, recall, and F1-score to ensure a reliable evaluation, especially in the presence of class imbalance.

III. Final Model

- **Selected Model: HDBSCAN & XGBoost**

After evaluating all models, **HDBSCAN & XGBoost** was selected as the final model for deployment due to its **superior performance** and **ability to capture complex attack patterns**.

- **Why This Choice?**

  - **HDBSCAN** naturally identifies groupings in data, allowing better structuring of attack categories.
  - **XGBoost**, an optimized decision tree-based model, refines these results by providing more precise supervised classification.
  - This model significantly outperformed the others, reaching **69.83% accuracy in testing and up to 91% in training**, although the **optimized hyperparameters could not be loaded** in the final application.

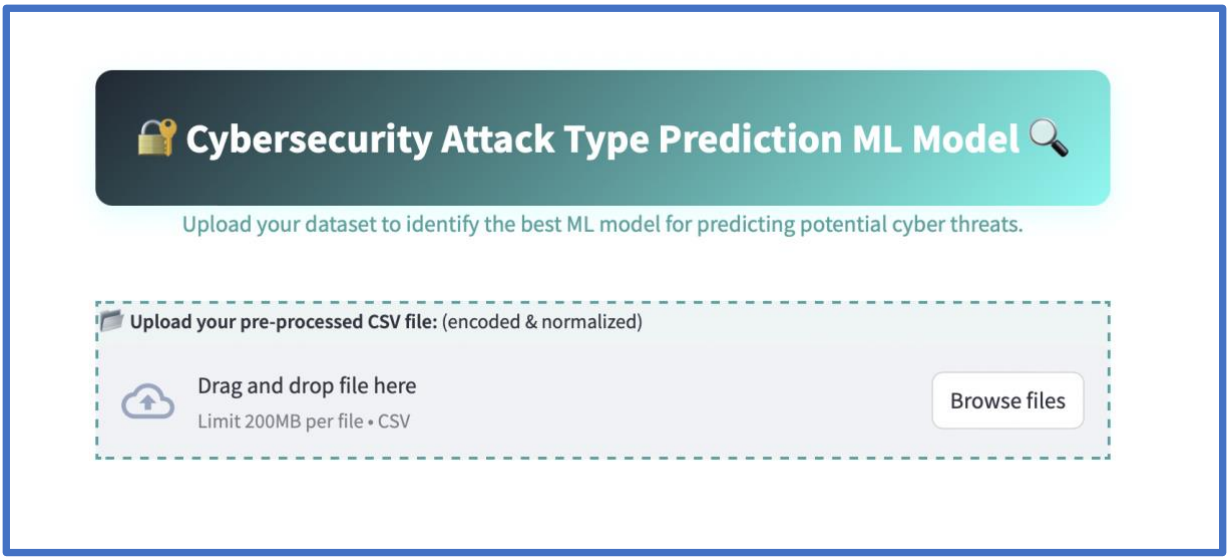
IV. Error Analysis

- **Limitations and Challenges Encountered**

  - **Issue with Hyperparameter OPTUNA Loading**

    - The optimized version of **XGBoost, which reached 91% accuracy**, could not be fully loaded into the application due to the **large model size**.
    - This limitation prevents fully utilizing the model's capabilities, although results remain promising with **69.83% accuracy in testing**.

3. Deployment of the Web Application



To provide users with an interactive and user-friendly way to access model predictions, we developed a web application using Streamlit.

The application allows users to upload a CSV file containing pre-encoded and normalized data, ready to be processed by the selected models: Logistic Regression, K-Nearest Neighbors, HDBSCAN, and XGBoost.
Once the file is uploaded, the application reads and displays the dataset using st.dataframe(), providing an overview of the data. The predictions from each model, along with their respective accuracy scores, are displayed in separate sections that users can access dynamically, ensuring a clear and structured visualization.

To make the application accessible without requiring local installations, we deployed it using Streamlit Community Cloud. This deployment enables users to access and interact with the application directly from their web browser, ensuring ease of use while maintaining a smooth and efficient workflow.

Conclusion

To conclude these in-depth analyses and numerous experiments involving several models, including Logistic Regression, K-Nearest Neighbors (KNN), HDBSCAN, and HDBSCAN & XGBoost, various approaches were tested. With accuracy levels of 34.49%, 33.51%, 49.81%, and 69.83%, respectively, the HDBSCAN & XGBoost model proved to be the best performer, making it the most suitable candidate for deployment.

Although the optimized hyperparameters could not be integrated into the final application due to technical constraints, the training results reached an accuracy of 91%, demonstrating the model's potential for improvement. However, such high accuracy during training presents a risk of overfitting, meaning that the model may adapt too closely to the training data and fail to generalize well to real-world data. Implementing stricter regularization and optimizing hyperparameters more effectively via OPTUNA—particularly **max_depth, min_child_weight, learning_rate, and num_boost_round**—could mitigate this issue and enhance the model's robustness.

To further improve the project's performance, several areas for enhancement can be explored. A new sidebar menu on the left could be introduced, allowing users to navigate between different sections, such as the **model accuracy and prediction** page, which is already in place. Additionally, we could enable users to select specific inputs such as **device** and **location** to refine predictions.

Based on the key characteristics selected in the **"cluster_model"** Jupyter notebook, where the application currently displays the predicted attack type (e.g., DDoS), we could integrate a **model selection** feature. This would allow the system to dynamically choose the most appropriate approach depending on the context. Consequently, the result could indicate whether the detected attack is **DDoS, Intrusion, or Malware**, inferred either through our best-performing model or a model manually selected by the user.

By implementing these improvements, the project could evolve into an advanced, high-performance solution for cyberattack detection and analysis, significantly strengthening cybersecurity on a large scale.