Detection and Classification of Network Attacks via Machine Learning

Abstract

In a context where cyberattacks are becoming increasingly sophisticated and varied, the detection and classification of network threats are major issues for the security of computer systems. This article explores the use of machine learning techniques applied to the CIC-UNSW-NB15 dataset to improve the efficiency of Intrusion Detection Systems (IDS). We address the challenges related to the diversity and volume of network data, and present supervised and unsupervised approaches to identify and classify various types of attacks, including DoS, backdoors, and exploits. The results obtained highlight the performance of individual and ensemble models (Stacking, Voting, Bagging) for precise and robust detection of network anomalies. Finally, we discuss the prospects offered by advanced techniques such as hyperparameter optimization and dimensionality reduction to enhance resilience against emerging threats.

1. Introduction

In an increasingly connected world, where the digitization of critical systems is becoming omnipresent, computer networks are constantly exposed to a growing variety of cyberattacks. These attacks, often sophisticated and ever-evolving, compromise not only the confidentiality, integrity, and availability of data, but can also disrupt critical infrastructures such as energy, financial, or health networks.

Among the most common forms of attacks are denial of service (DoS) attacks, malware such as backdoors, exploitations of vulnerabilities, and reconnaissance activities aimed at identifying flaws in systems. Effective detection and classification of these attacks are therefore essential to prevent damage and ensure the resilience of networks.

However, the challenges to be addressed are numerous:

- Massive volume of network data: Modern systems generate unprecedented volumes of data, making manual analysis impractical.
- Complexity and diversity of attacks: New threats frequently emerge, rendering signature-based or static rule solutions obsolete.

In the face of these challenges, machine learning models offer a promising alternative. They are capable of:

1. Identifying complex patterns in large datasets.
2. Adapting to new threats without requiring extensive reprogramming.

This project thus aims to explore machine learning techniques to improve the detection and classification of network attacks, focusing on efficiency, accuracy, and adaptability in the face of ever-evolving threats. By relying on the dataset CIC-UNSW-NB15, this work highlights the importance of both supervised and unsupervised approaches in the field of cybersecurity.

ORCID(s):

2. Methodology

2.1. Dataset Presentation

The dataset used for this project is the **CIC-UNSW-NB15**, a dataset designed for the evaluation of intrusion detection systems and the classification of network attacks. It contains detailed information about simulated network flows and covers a wide variety of attack scenarios and legitimate traffic. The data is divided into two main files: `Data.csv` and `Label.csv`.

File Structure

1. `Data.csv` :
   - This file contains the features extracted from the network flows. The columns include measures such as flow durations, statistics on transmitted packets (average, maximum, minimum size), temporal indicators, and flags specific to network protocols.
   - The 78 columns present can be grouped into several main categories:
     - **Temporal Features**: Flow durations (`Flow Duration`), packet intervals (`Flow IAT Mean`, `Flow IAT Max`), etc.
     - **Packet Statistics**: Sizes of transmitted packets (`Packet Length Mean`, `Packet Length Max`), variance and distribution, etc.
     - **Flow-specific Indicators**: TCP flags (`SYN Flag Count`, `ACK Flag Count`), traffic reports (`Flow Bytes/s`, `Flow Packets/s`).
     - **Advanced Features**: Statistics on sub-flows (`Subflow Fwd Packets`, `Subflow Bwd Packets`) and indicators specific to the beginning of connections (`FWD Init Win Bytes`, `Bwd Init Win Bytes`).
   - Each line of the file corresponds to a unique network flow.
2. `Label.csv` :
   - This file provides the labels associated with each network flow. The labels are numeric values indicating the type of traffic, whether it is legitimate or malicious.
   - The classes are defined as follows (according to the Readme file):
     - **0**: Benign traffic (normal).
     - **1**: Scanning (scans and reconnaissance).
     - **2**: Backdoor.
     - **3**: Denial of Service (DoS) attacks.
     - **4**: Exploits (exploitation of vulnerabilities).
     - **5**: Fuzzers (automated vulnerability testing).
     - **6**: Generics (generic cryptographic attacks).
     - **7**: Reconnaissance.
     - **8**: Shellcode.
     - **9**: Worms (computer worms).

Size and Diversity

The CIC-UNSW-NB15 dataset has been configured with a distribution of 80% benign data and 20% malicious data corresponding to attacks.

- **Total number of connections**: The `Data.csv` file contains a total of 447,915 network flows.
- **Number of features**: 76 in `Data.csv` and 1 in `Label.csv`
- **Distribution of connections by type**:
  - **0: Benign traffic (normal)**: 358,332 connections.
  - **1: Scanning (scans and reconnaissance)**: 385 connections.
  - **2: Backdoor**: 452 connections.
  - **3: Denial of Service (DoS) attacks**: 4,467 connections.
  - **4: Exploits (exploitation of vulnerabilities)**: 30,951 connections.
  - **5: Fuzzers (automated vulnerability testing)**: 29,613 connections.
  - **6: Generics (generic cryptographic attacks)**: 4,632 connections.
  - **7: Reconnaissance**: 16,735 connections.
  - **8: Shellcode**: 2,102 connections.
  - **9: Worms (computer worms)**: 246 connections.

Importance for the Project

The dataset **CIC-UNSW-NB15** offers a wealth of realistic scenarios and attacks, making it ideal for:

- Evaluating the ability of learning models to detect and classify different types of threats.
- Testing the effectiveness of both supervised and unsupervised approaches on diverse network data.

By leveraging this dataset, the project aims to develop models capable of quickly detecting anomalies and accurately classifying types of attacks in a network environment.

2.2. Data Preprocessing

The `CIC-UNSW-NB15` dataset underwent a series of preliminary treatments to ensure its quality and relevance for the analysis. These steps include handling missing values, deduplication, column transformation, and the removal of unnecessary features.

Adding and Transforming Columns

- A `Label Num` column was added to store the numeric values of the labels, identifying the types of connections (benign or malicious).
- An additional column, `Flow Duration (seconds)`, was created by converting flow durations from microseconds to seconds to facilitate interpretation and analysis.
- A `Connection Type` column was generated to distinguish benign connections (*Benign*) from attacks (*Attack*) based on the labels.

Handling Missing Values and Data Types

- No missing values were detected in the dataset, as confirmed by exploratory analysis (`data.isna().sum()`). Therefore, no corrective actions were necessary.
- The data types were also checked, and all fields were in the expected format (e.g., `int64` or `float64`). Thus, no further conversion was necessary.

Removing Duplicates

- The initial dataset contained 447,915 rows. After removing duplicates, 306,173 rows were retained, corresponding to a reduction of 141,742 rows (approximately 32% of the dataset). This step helped eliminate redundant information and reduce the size of the data for more efficient processing.

Removing Non-informative Columns

Some columns were identified as non-informative because they had identical values for all records. These columns were removed to avoid introducing noise in the learning models. The following columns were eliminated:

- `Bwd PSH Flags`
- `Fwd URG Flags`
- `Bwd URG Flags`
- `URG Flag Count`
- `CWR Flag Count`
- `ECE Flag Count`
- `Fwd Bytes/Bulk Avg`
- `Fwd Packet/Bulk Avg`
- `Fwd Bulk Rate Avg`

Summary of Preprocessing Steps

After these steps, the final dataset consists of 306,173 records and 77 columns. These adjustments ensure a dataset ready for in-depth analysis and effective modeling.

2.3. Exploratory Data Analysis (EDA)

2.3.1. Distribution of connections: Benign vs Attacks

After the preprocessing steps, the proportion between benign and malicious connections slightly evolved, changing from an initial distribution of 80% for benign flows and 20% for attacks to a distribution of 73.5% for benign flows and 26.5% for attacks (see Figure 1). This change is mainly due to the removal of duplicates and non-informative columns. Despite this variation, benign flows remain overwhelmingly majority, illustrating a class imbalance while still reflecting the distributions observed in real-world scenarios.

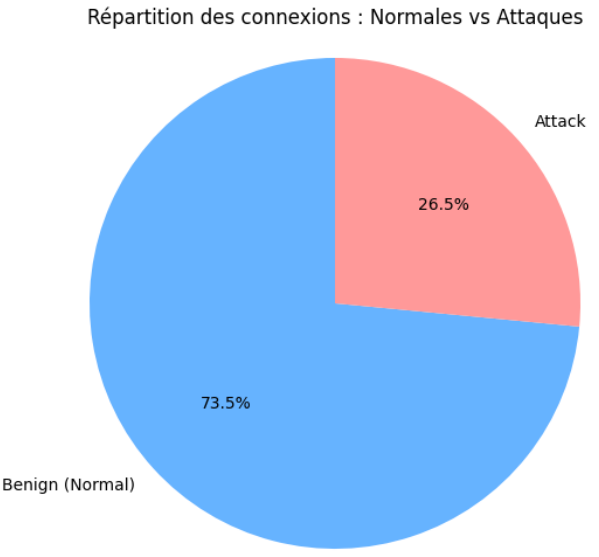


Figure 1: Distribution of connections: Benign vs Attacks.

2.3.2. Distribution of Attack Types

This analysis highlights the frequency of the various categories of attacks present in the dataset after preprocessing. The most represented attack categories are *Exploits* (30,951 occurrences) and *Fuzzers* (29,613 occurrences), which together constitute a significant proportion of the malicious traffic (see Figure 2). In contrast, some categories, such as *Worms* (246 occurrences) and *Analysis* (385 occurrences), are markedly underrepresented. This underrepresentation could lead to a decrease in the performance of machine learning models in classifying these types of attacks.

These variations in the frequency of attacks likely reflect realistic scenarios, where certain forms of attacks are more widespread or easier to simulate in controlled environments. These results underscore the importance of using robust modeling strategies capable of managing class imbalance. For example, techniques such as oversampling rare classes or undersampling majority classes could be employed to improve model performance.

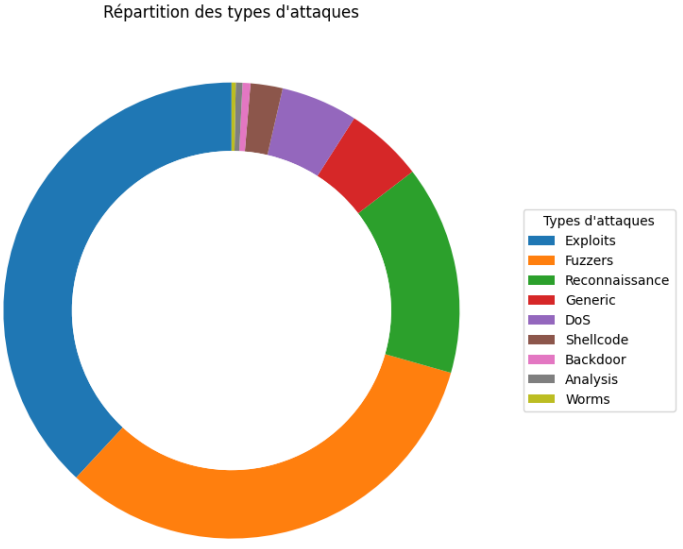


Figure 2: Distribution of attack types

2.3.3. Flow Duration by Connection Type (Box Plot)

The analysis reveals that, although certain attack categories like *Exploits* and *Fuzzers* show a slightly wider dispersion of their durations, it is clear that flow duration, when considered in isolation, is not a sufficient criterion to distinguish malicious flows from benign flows. For example:

- Benign connections, while concentrated around short durations, share similar value ranges with several types of attacks.
- The durations of attacks, such as *DoS*, *Reconnaissance*, and even *Worms*, significantly overlap with those of benign flows, making it difficult to make a clear distinction based solely on this characteristic.

These results highlight the limitation of flow duration as a discriminating criterion. To improve the differentiation between normal and malicious traffic, it is necessary to integrate other criteria, such as packet volume, packet size, or behavioral indicators.

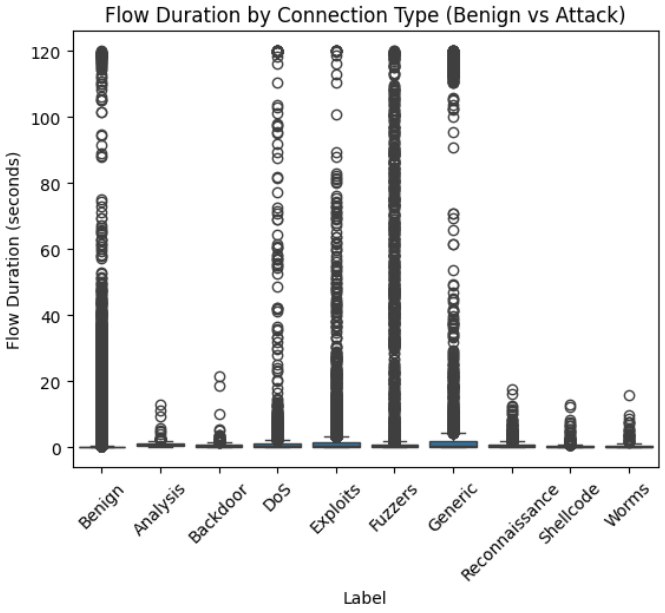


Figure 3: Flow duration by connection type.

2.3.4. Relation between flow duration and connection types (Normal vs Attack)

This visualization illustrates the relationship between network flow duration (*Flow Duration*) and the total number of packets transmitted forward (*Total Fwd Packet*) and backward (*Total Bwd Packet*) according to the type of connection (attack or benign).

Malicious connections (in red) show greater variability, including flows with high durations and packet volumes, while benign connections (in blue) primarily cluster around shorter durations and limited volumes. These marked differences between benign and malicious flows highlight exploitable patterns for anomaly detection and network attack recognition.

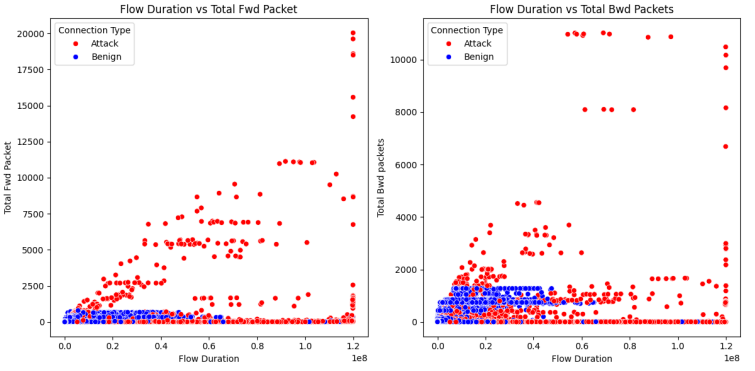


Figure 4: Relationship between flow duration and transmitted packets (*Total Fwd Packet* and *Total Bwd Packet*) according to the type of connection.

Furthermore, by combining the flow duration with the total number of transmitted packets (*Total Fwd Packet* and *Total Bwd Packet*), it becomes possible to better distinguish malicious connections from benign ones. For example:

- Malicious connections exhibit extreme packet volumes and increased variability compared to benign connections.
- Benign flows, on the other hand, remain concentrated within short duration ranges and more predictable packet volumes, allowing for clearer separation when another parameter is added to the analysis.

These observations reinforce the importance of a multi-parameter approach to effectively capture the behaviors of malicious network flows in a complex environment.

2.3.5. Heatmap of Correlations

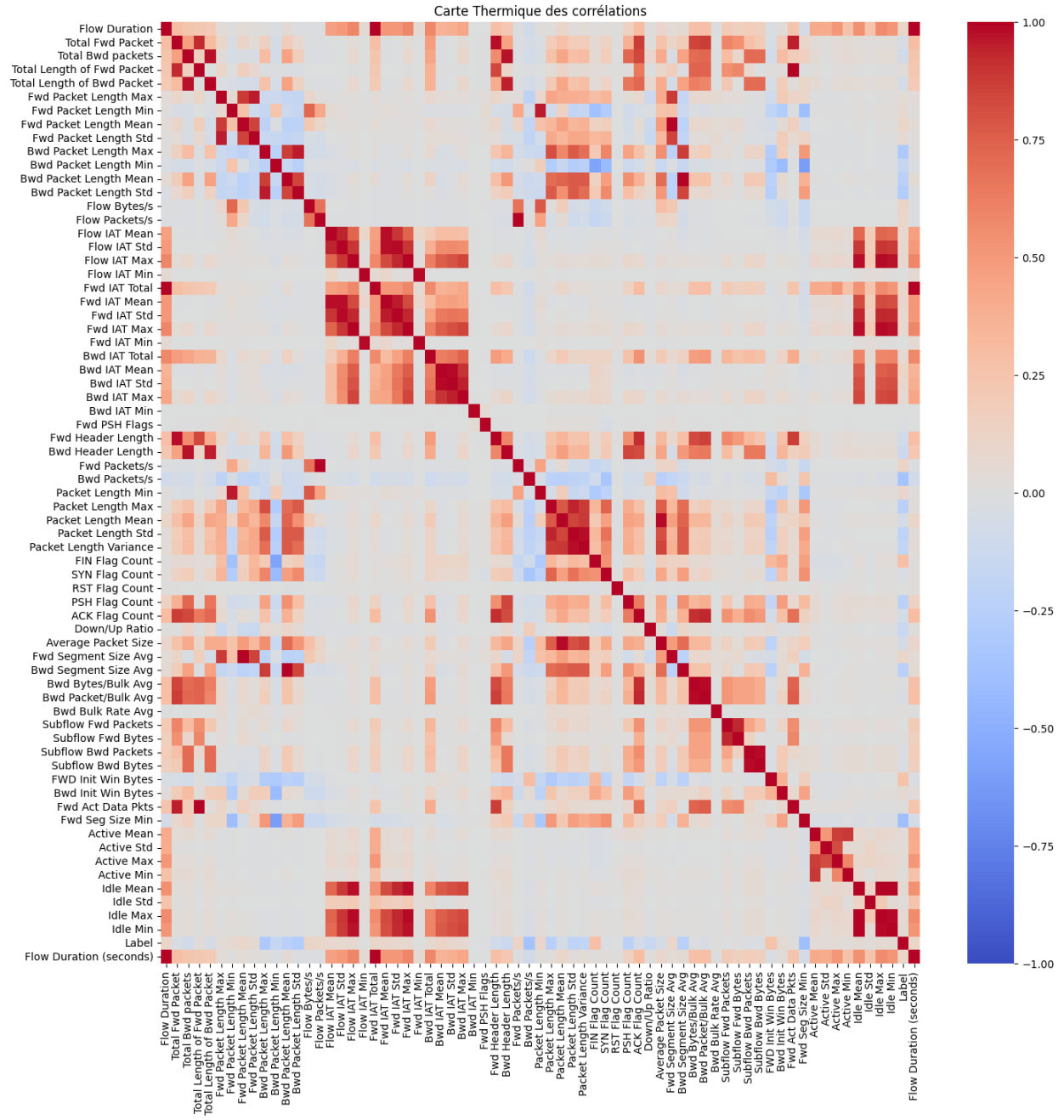


Figure 5: Heatmap of correlations between the different characteristics of network flows.

Analysis and Interpretations:

- **Strong Positive Correlations:** Some features show very high positive correlations, indicated by dark red cells. For example:
  - Temporal features such as *Flow IAT Mean* and *Flow IAT Std* are strongly correlated, reflecting an intrinsic relationship between the mean and the dispersion of the intervals between packet arrivals in a flow.
- **Significant Negative Correlations:** Significant inverse relationships can be identified, represented by dark blue cells. For example:
  - *Bwd Packet Length Min* and *Fwd Act Data Pkts* exhibit a notable negative correlation. This indicates that a decrease in the minimum length of backward packets is often associated with an increase in the number of packets containing active data forwards, which may reflect an asymmetry in network flow behavior.
- **Weak Correlations:** Some variables, such as flag indicators (*FIN Flag Count*, *RST Flag Count*), show weak correlations with other features. This indicates that these variables might provide complementary and independent information to learning models, particularly in nonlinear or unsupervised approaches.

Perspectives for Machine Learning:

- **Dimensionality Reduction:** Features that show very high correlations, such as *Packet Length Mean* and *Packet Length Std*, can be considered redundant. Their removal or grouping (for example, using methods such as Principal Component Analysis, PCA) would help reduce model complexity while maintaining a large portion of the relevant information.
- **Selection of the Most Relevant Features:** The relationships identified with variables such as *Flow Duration*, *Flow IAT Mean*, and *Packet Length Variance* underscore their importance for anomaly detection and flow classification. These features capture critical behaviors of network flows and should be included in the models.

2.3.6. Anomaly Detection with Isolation Forest

The *Isolation Forest* approach is utilized to identify anomalies in the dataset by classifying flows as normal or abnormal. The graph on the right (Figure 6) illustrates the performance of this method by differentiating benign and malicious flows. It presents the separation between normal flows (in blue) and detected anomalies (in red), based on flow duration (*Flow Duration*) and the total number of packets forwarded (*Total Fwd Packet*). The results show a concentration of normal flows around short durations and a low volume of packets, while anomalies span a wider range of both variables. This illustrates the effectiveness of the Isolation Forest approach to detect unusual behaviors.

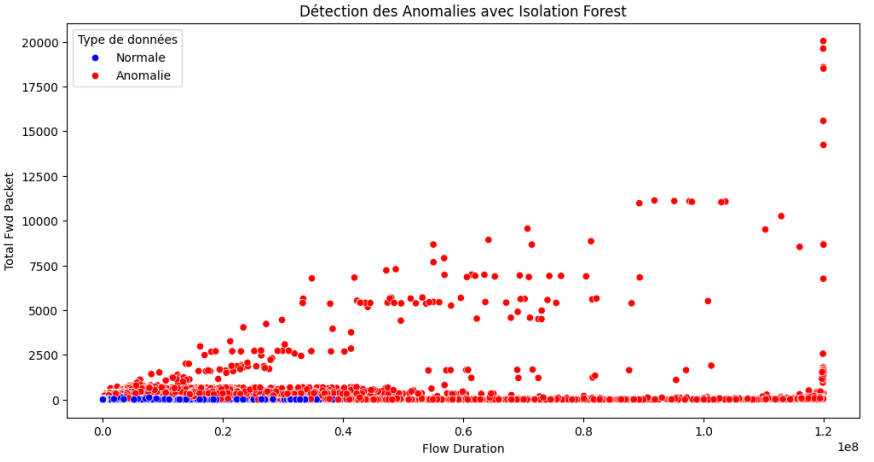


Figure 6: Anomaly detection with Isolation Forest.

The second graph (Figure 7) refines this analysis by incorporating information about malicious and benign flows. It shows that undetected malicious flows (orange points) represent a critical error for attack detection, as these flows go unnoticed and evade any protective measures. Conversely, the pink points represent benign flows misclassified as anomalies, indicating false positives. Although these are less critical, they nonetheless reveal limitations in the algorithm's ability to distinguish normal flows from malicious flows. These results highlight the challenges posed by the overlap of features between benign and malicious flows, complicating classification and underscoring the need for complementary approaches to improve overall accuracy and prioritize the reduction of false negatives.

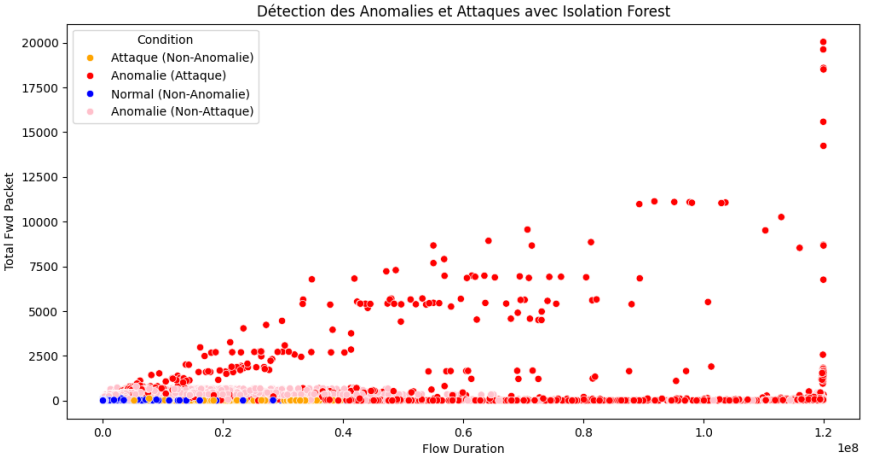


Figure 7: Classification of flows according to their condition (attack or normal) with Isolation Forest.

2.3.7. Exploration of Relationships between Dataset Features

This section focuses on the in-depth analysis of relationships between certain features of the dataset, in order to assess their potential to discriminate between normal flows and attacks, and to improve the classification of specific types of attacks. Initially, we will examine the features that differentiate benign connections from attacks. Then, we will analyze the key features that help distinguish between different categories of attacks.

Differentiation of Benign Connections and Attacks:

Total number of packets transferred by connection type: The analysis of the total number of packets transferred reveals a clear distinction between benign connections and attacks. Benign traffic shows a significantly higher volume of packets transferred than all combined categories of attacks. This behavior is representative of regular and continuous connections, such as those observed in normal scenarios.

In contrast, attacks, especially *Exploits*, *DoS*, and *Fuzzers*, exhibit much lower packet volumes. This reflects irregular behaviors or those specific to targeted malicious actions. For example:

- *DoS* attacks generate concentrated traffic over a short duration, but their total volume remains lower than that of benign flows.
- *Exploits* attacks display a more moderate volume, corresponding to repeated but localized exploitation attempts.

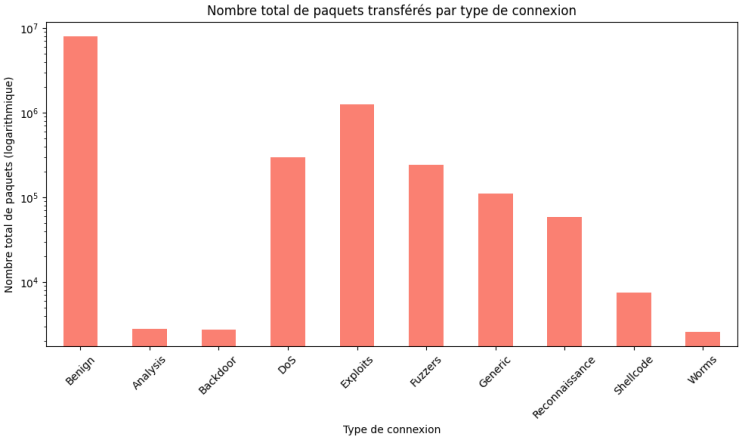


Figure 8: Total number of packets transferred by connection type.

These observations highlight the importance of the total number of packets transferred as a key distinguishing feature for separating benign traffic from attacks. Benign flows, with their high and continuous volume, can be quickly identified, while attacks cluster in ranges of lower volumes. This distinction can be effectively used in anomaly detection models for initial traffic classification.

Total number of packets received by connection type: The analysis of the total number of packets received also reveals a significant difference between benign connections and attacks. Benign traffic (*Benign*) presents a markedly higher volume of received packets than all categories of attacks combined. This reflects the stable and sustained nature of normal flows.

In contrast, attacks, particularly *Exploits*, *DoS*, and *Fuzzers*, show much lower volumes of received packets. These differences can be explained by the characteristics of specific attacks:

- *DoS* attacks focus on overwhelming resources by issuing massive requests, but they do not necessarily involve a significant return of data.
- *Exploits* attacks tend to generate targeted exchanges with a limited number of packets, reflecting specific exploitation attempts.

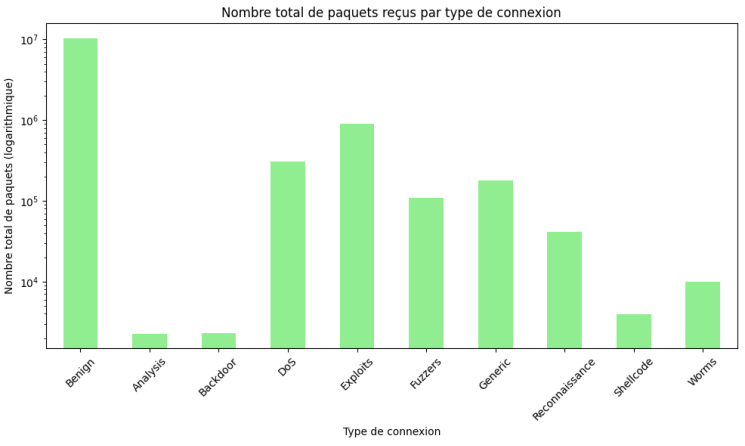


Figure 9: Total number of packets received by connection type.

These results reinforce the idea that the volume of packets received is a key characteristic for differentiating benign flows from malicious flows. Machine learning models can leverage these observations to enhance anomaly detection accuracy and classify types of network flows.

Average duration of high-bandwidth connections (Bytes/s): The analysis of the average duration of high-bandwidth connections highlights a clear separation between benign traffic (*Benign*) and network attacks.

The following observations stand out:

- Benign traffic has a significantly higher average connection duration than all categories of attacks combined. This characteristic reflects regular and continuous network behaviors associated with stable data transfers over extended periods.
- Attacks, such as *Exploits*, *DoS*, and *Generic*, exhibit significantly shorter durations, corresponding to sporadic or intensive behaviors over a limited time.
- The attack categories *Reconnaissance*, *Shellcode*, and *Worms* are characterized by particularly low durations, indicating rapid and targeted actions within the network.

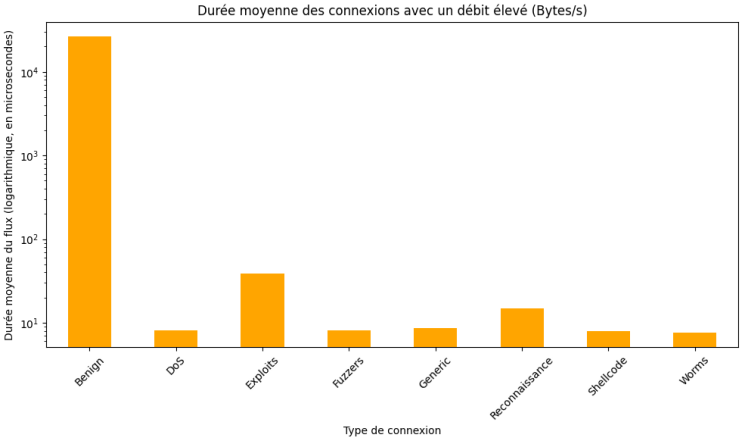


Figure 10: Average duration of high-bandwidth connections (Bytes/s).

These results emphasize the importance of average duration combined with high bandwidth as a discriminating feature for differentiating normal connections from malicious ones. This distinction is especially useful for identifying abnormal behaviors in high-bandwidth systems where connection stability plays a critical role.

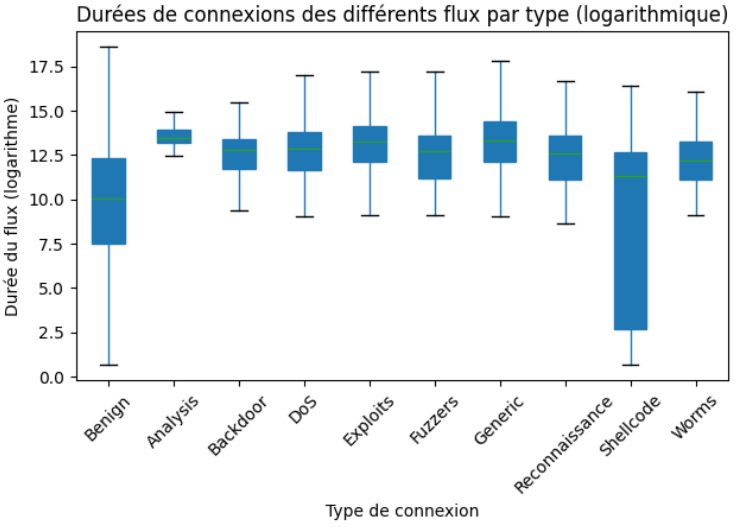
Differentiation Between Attack Types:

Average flow duration by connection type and detailed analysis: The analysis of the average flow duration reveals significant differences between the various types of attacks. Connections associated with *Generic* attacks exhibit exceptionally high average durations compared to other categories, distinctly differentiating them from other types of attacks. This may reflect scenarios where prolonged flows are used for specific tasks, such as large-scale data transfers or ongoing exploitation processes.

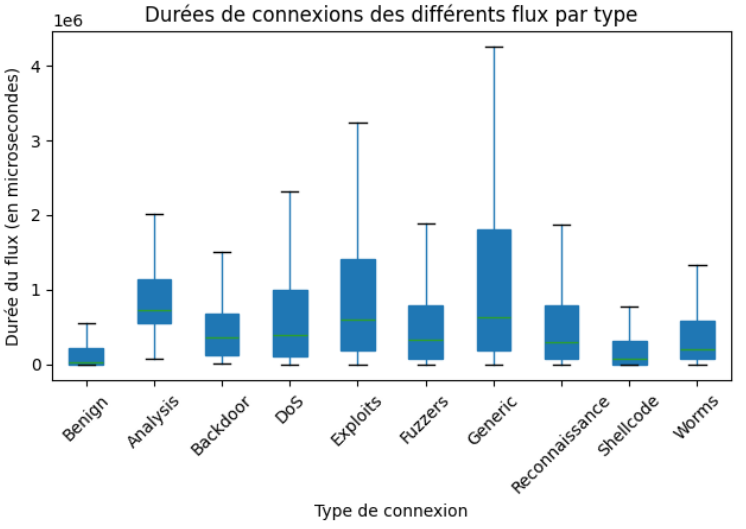
In comparison, *DoS* and *Exploits* attacks show relatively short flow durations, consistent with their nature:

- *DoS* attacks generally involve a large number of short connections intended to overload the resources of a target system.
- *Exploits* attacks consist of targeted and repeated attempts, often with limited durations to evade detection.

Benign flows (*Benign*), on the other hand, display average durations lower than most attacks, reflecting their stable and non-intrusive nature. These differences in flow duration can be leveraged to enhance the classification of attack types by incorporating this feature into machine learning models to capture atypical behaviors related to connection durations.



(a) **Logarithmic flow duration by connection type.**



(b) **Flow duration by connection type (linear).**

Figure 11: Analysis of flow duration by connection type: logarithmic and linear.

Average connection throughput (Bytes/s) by connection type: The analysis of the average connection throughput (Bytes/s) highlights significant differences between attack types and benign traffic (*Benign*). Benign connections exhibit a very low average throughput, characteristic of stable and regular network traffic. In contrast, certain categories of attacks, such as *Generic*, *Shellcode*, and *Worms*, are distinguished by particularly high average throughput, suggesting intensive data transfers over a limited period.

In detail:

- The *Generic* attacks exhibit the highest average throughput, reflecting massive data transfer activities, often linked to attempts at exfiltration or resource overload.
- *Shellcode* and *Worms* attacks also show significant throughput, which may indicate aggressive propagation or execution of malicious payloads.

These variations in average throughput can be exploited as a key indicator to distinguish not only attacks from benign flows but also to differentiate between attack types. Their integration into machine learning models can enhance algorithms' ability to identify abnormal behaviors in real time.

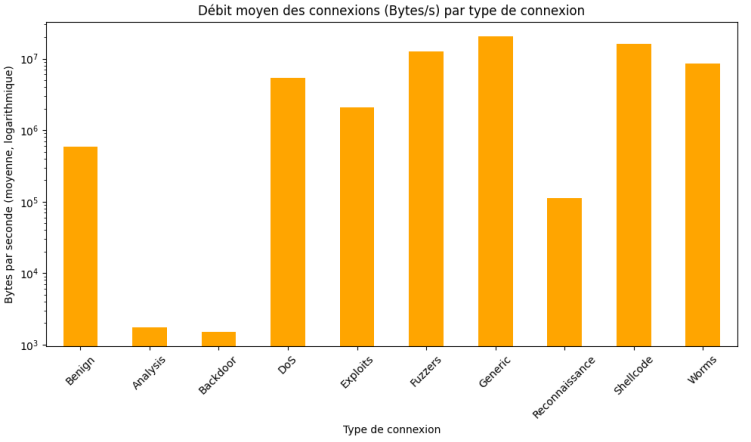


Figure 12: Average connection throughput (Bytes/s) by connection type.

Total number of packets transferred for long-duration flow attacks: The analysis of the total number of packets transferred for long-duration flow attacks reveals characteristic patterns that can help distinguish between different categories of attacks.

In particular:

- *Exploits* attacks clearly dominate in terms of transferred packet volume, reflecting repeated exploitation attempts over an extended duration.
- *DoS* attacks also present a high volume of packets, although more concentrated, indicating intense actions over a relatively short period.
- *Fuzzers* and *Generic* attacks, on the other hand, show moderate volumes, highlighting more localized or specific behaviors.

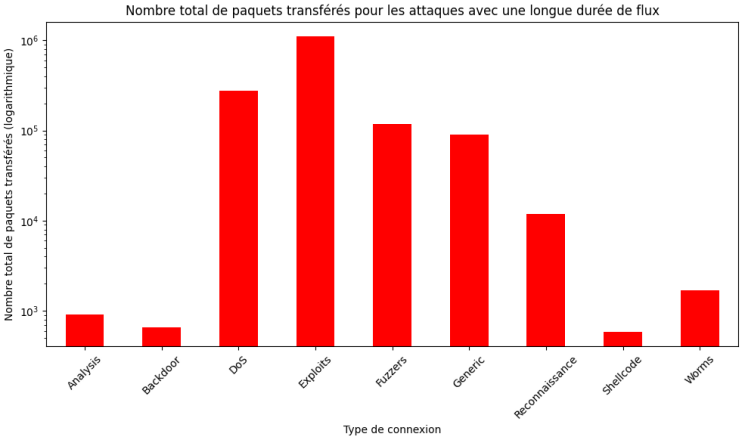


Figure 13: Total number of packets transferred for long-duration flow attacks.

These observations underscore the diversity of strategies employed in long-duration network attacks, where certain categories exploit massive data volumes to overwhelm systems (e.g., *DoS*), while others favor more subtle but prolonged approaches (e.g., *Exploits*).

The integration of this feature into machine learning models can be crucial for identifying these prolonged attacks and anticipating their potential impact on network infrastructures.

Top 100 attack connections with the most packets received: The analysis of the 100 attack connections that received the highest number of packets reveals distinct characteristics among the different categories of attacks.

The main results are as follows:

- *DoS* attacks clearly stand out with a significantly higher volume of packets received compared to other categories. This behavior is expected, as *DoS* attacks generally involve an intense and sustained flow aimed at overwhelming target resources.
- *Exploits* and *Generic* attacks also show high volumes of packets received, although less pronounced than the *DoS*. This reflects attack tactics requiring multiple exchanges to exploit vulnerabilities or carry out prolonged malicious actions.
- Other categories, such as *Fuzzers*, *Reconnaissance*, and *Shellcode*, exhibit much lower volumes, suggesting more targeted or limited actions in terms of network interaction.

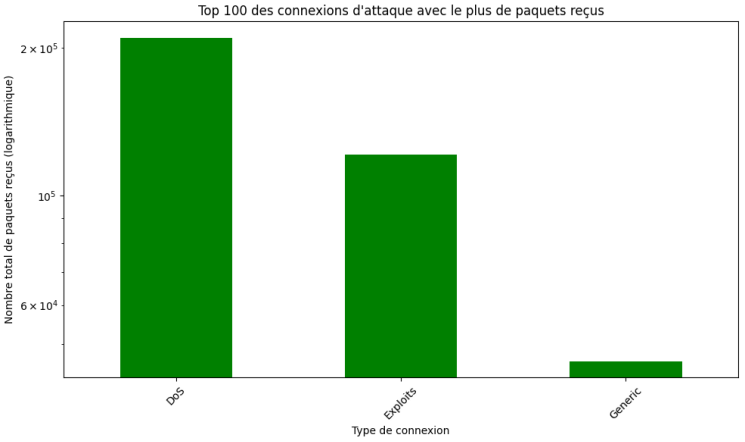


Figure 14: Top 100 attack connections with the most packets received.

These observations highlight the importance of connections with a large number of received packets as indicators for identifying certain types of attacks, particularly *DoS*. This feature can be utilized to prioritize the detection and classification of the most harmful network attacks in a real-time context.

Conclusion : The analysis of the relationships between the different characteristics of the dataset highlighted key variables that allow for the distinction between benign connections and attacks, as well as the differentiation of attack types from one another. Among these characteristics, the total number of packets transferred and received, the average duration of flows, the average throughput of connections, and the volumes of data exchanged in specific scenarios (long durations or intense flows) proved to be particularly discriminative.

These observations emphasize the following points:

- Benign connections are characterized by high and steady volumes, prolonged durations, and generally low throughputs, reflecting regular and predictable behaviors.
- Attacks, on the other hand, show varied behaviors: *DoS* and *Generic* attacks are associated with high volumes or intense throughputs, while *Exploits* and *Fuzzers* are characterized by shorter durations and more moderate volumes.
- Certain specific categories, such as *Shellcode*, *Worms*, and *Reconnaissance* attacks, stand out due to extreme or targeted characteristics, thus providing unique clues for their identification.

3. Implementation and Results

3.1. Dimensionality Reduction

3.1.1. PCA (Principal Component Analysis)

Principal Component Analysis (PCA) was performed to reduce the dimensionality of the dataset while retaining most of the variance. This method allows for the visualization of data in a reduced-dimensional space and helps to detect groupings or anomalies.

Visualization of Data after PCA (2D): In Figure 15, the data projected onto the first two principal components (PC1 and PC2) show some separation between types of attacks and benign traffic (*Benign*). The *DoS*, *Exploits*, and *Generic* attacks exhibit partial groupings, while the other categories mix more, suggesting a stronger similarity among their main characteristics.

However, this visualization also highlights significant overlap between certain classes, particularly benign flows and some attacks (*Reconnaissance*, *Worms*), making their differentiation more challenging.

Visualization after Feature Selection (2D): Figure 16 shows the data after selecting the most relevant features before applying PCA. This step allowed for better separation of the data, but their organization remains relatively cluttered, still making it difficult to identify clear structures among the different classes.

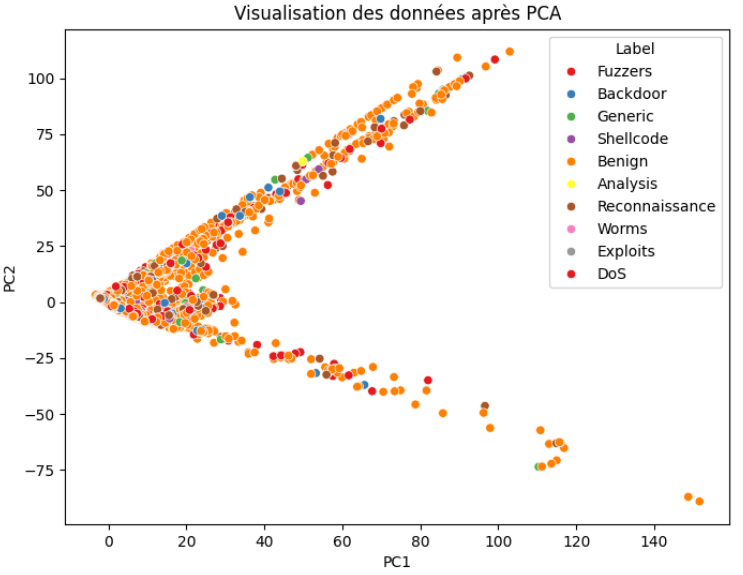


Figure 15: Visualization of data after PCA (2D).

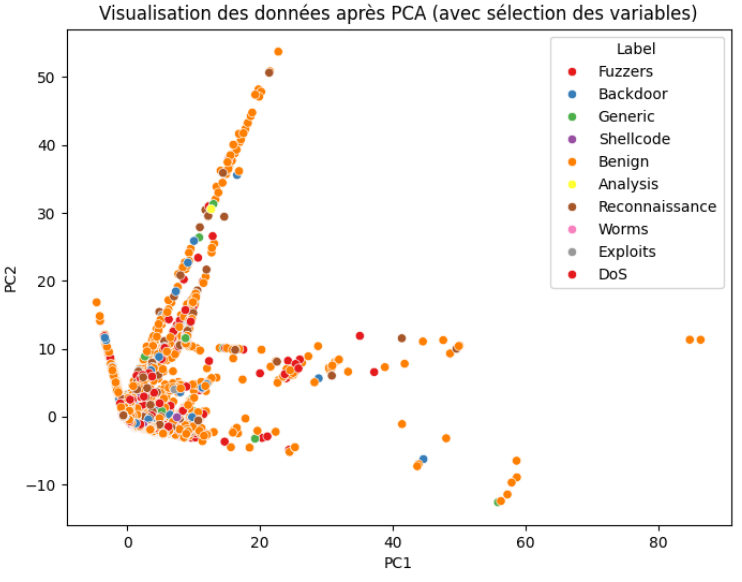


Figure 16: Visualization of data after PCA with feature selection.

Visualization of Data in 3D after PCA: Figure 17 presents a 3D projection of the data onto the first three principal components (PC1, PC2, and PC3). However, this visualization does not provide significant information compared to the other two-dimensional representations. The groupings remain vague, and the overlaps between the different data categories, especially between attacks and benign traffic, persist strongly. This graph illustrates the limitations of PCA in differentiating classes in complex datasets.

Conclusion: The application of PCA allowed for a reduction in dimensionality while preserving variance. Nevertheless, the visualizations obtained, whether in 2D or 3D, do not allow for a clear-cut separation of categories, particularly for benign flows and certain attacks (*Reconnaissance*, *Worms*).

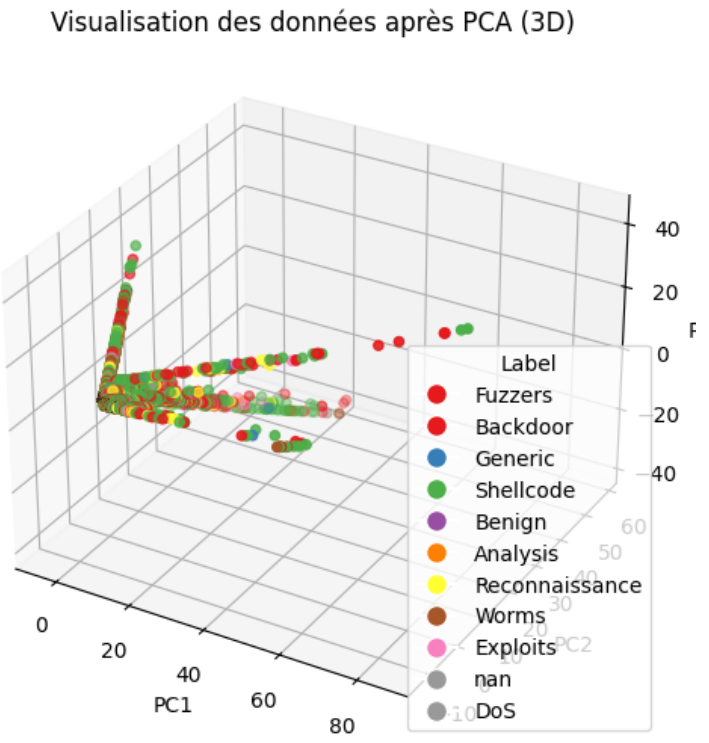
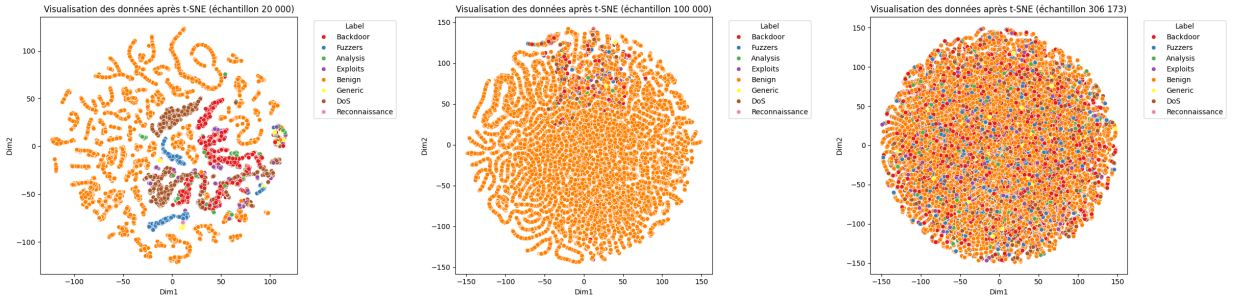


Figure 17: Visualization of data after PCA (3D).

3.1.2. t-SNE

The *t-Distributed Stochastic Neighbor Embedding* (t-SNE) technique has been used to reduce the dimensionality of the data and facilitate their visualization in a low-dimensional space (2D). This method is particularly useful for capturing local relationships between data points while preserving global structures.



(a) **Sample of 20,000 points** (b) **Sample of 100,000 points** (c) **Complete set (306,173 points)**

Figure 18: Visualization of data after dimensionality reduction with t-SNE for different samples.

The visualizations obtained using t-SNE show the evolution of groupings in the data according to the size of the analyzed sample. For a small sample of 20,000 points (Figure 18a), apparent groupings are observed, with a partial separation between benign flows and certain categories of attacks. Increasing the sample size to 100,000 points (Figure 18b) makes these groupings less distinct, indicating an increasing overlap between classes. Finally, when t-SNE is applied to the complete dataset (Figure 18c), it becomes clear that no significant grouping can be identified. This suggests that the complexity and diversity of the data make it difficult to differentiate categories on a large scale, requiring complementary approaches to analyze this data.

3.2. Supervised Machine Learning

3.2.1. Attack Identification (Individual Models)

In this step, several individual models were trained for attack identification using binary training data. The scores of the main metrics for each model (Precision, Recall, F1-Score, and Overall Precision) were compared, and the results are presented in Figure 19.

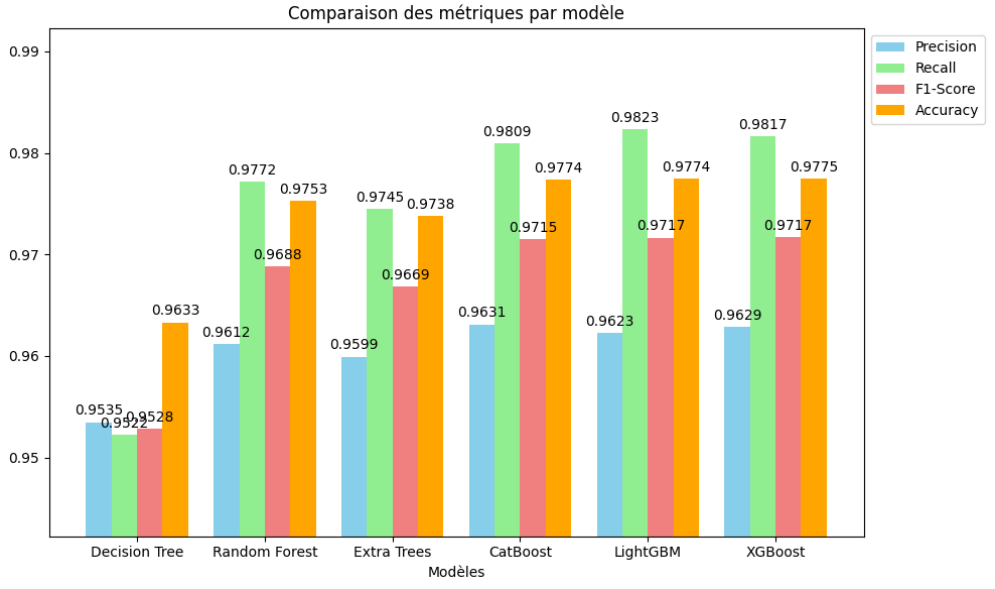
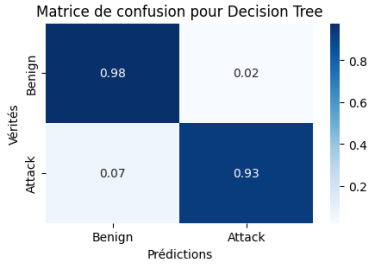
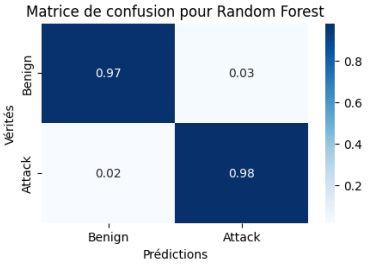


Figure 19: Comparison of metrics for different individual models.

The **XGBoost** model achieved the best scores, tying with **CatBoost** and **LightGBM** in terms of precision, recall, and F1-Score. However, XGBoost stood out due to its faster execution time, making it the optimal model for this task.

The confusion matrices for each model are presented below (Figures 20 to 21). They show similar rates of false negatives and false positives for the different models. However, **XGBoost** remains slightly superior, followed by **CatBoost** and **LightGBM**.





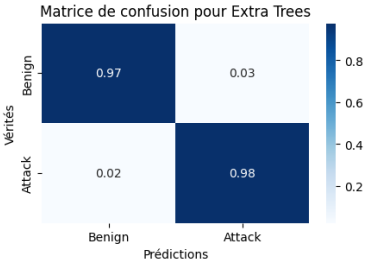


Figure 19: Decision Tree

Figure 19: Random Forest

Figure 20: Confusion matrices for Decision Tree, Random Forest, and Extra Trees.

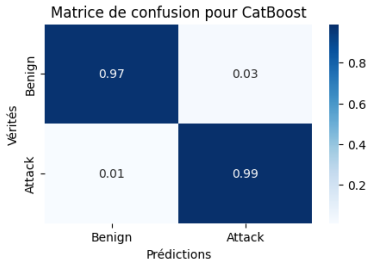


Figure 20: CatBoost

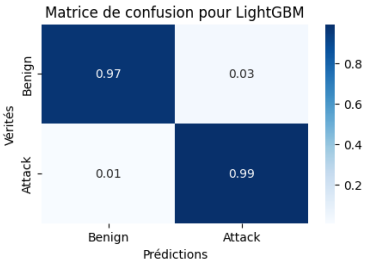


Figure 20: LightGBM

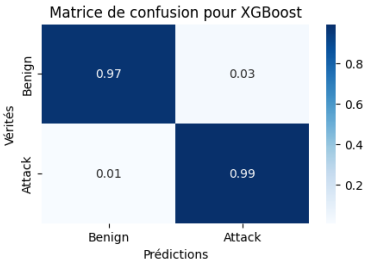


Figure 21: Confusion matrices for CatBoost, LightGBM, and XGBoost.

To further improve the performance of the **XGBoost** model, a search for optimal hyperparameters was conducted. The results after optimization are detailed below:

- **Overall Precision** : 0.9773
- **Overall Recall** : 0.98
- **Overall F1-Score** : 0.97
- **Class "Benign"** :
  - Precision : 1.00
  - Recall : 0.97
- **Class "Attack"** :
  - Precision : 0.93
  - Recall : 0.99

By comparing these results with those obtained before optimization, it is observed that:

- The overall precision remains stable (from 0.9775 to 0.9773), with a very slight decrease.
- The overall recall decreases slightly (from 0.9817 to 0.98), but remains high.
- The overall F1-Score showed a slight improvement (from 0.9717 to 0.97), indicating a better balance between precision and recall.
- For the "Benign" class, precision reaches a perfect value (1.00), reducing false positives (attacks incorrectly classified as benign).
- For the "Attack" class, although precision decreases (from 0.97 to 0.93), recall remains high (0.99), limiting false negatives (benign samples incorrectly classified as attacks).

These results confirm that hyperparameter optimization allows for a better balance in class classification, particularly by reducing errors for the "Benign" class.

In conclusion, **XGBoost**, with or without hyperparameter optimization, remains the most effective model for attack identification due to its balanced overall performance and fast execution time. However, the choice of final parameters will depend on the specific priorities of the application, such as minimizing false positives or false negatives.

3.2.2. Attack Classification (Individual Models)

In this step, several individual models were trained specifically for the classification of different categories of attacks. The main metrics such as precision, recall, F1-score, and overall accuracy were compared for each model. The results are presented in Figure 22.

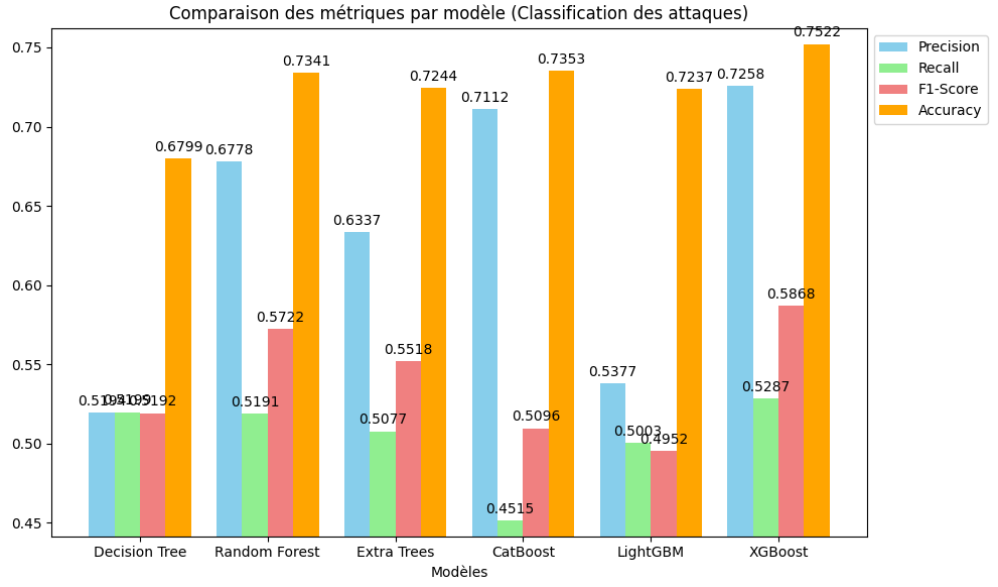


Figure 22: Comparison of metrics by model for attack classification.

As observed in the results, the **XGBoost** model continues to stand out, closely followed by **LightGBM** and **CatBoost**. XGBoost achieves the best scores in terms of F1-score and overall accuracy, making it particularly effective for multi-class classification of attacks. However, models such as **Decision Tree** and **Extra Trees** show inferior performance, particularly on less represented classes.

The confusion matrices for each model (Figures 23 to 24) highlight the specific strengths and weaknesses of the models in classifying the different categories of attacks. For example, the **XGBoost** and **LightGBM** models display high detection rates for frequent classes such as *Exploits* and *Fuzzers*, but show slightly weaker performance for rarer classes like *Worms* and *Shellcode*.

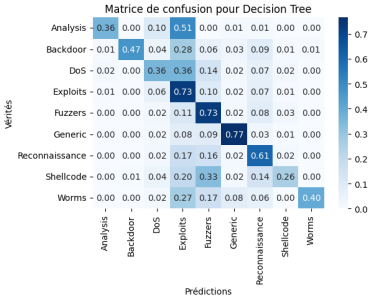


Figure 22: Decision Tree

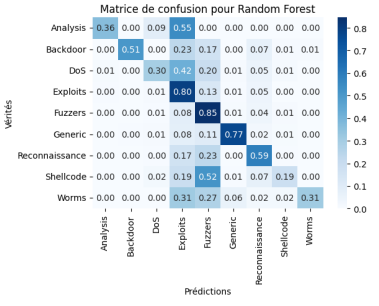


Figure 22: Random Forest

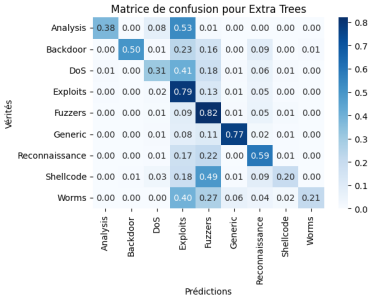


Figure 23: Confusion matrices for Decision Tree, Random Forest, and Extra Trees in attack classification.

In conclusion, although all models show acceptable performance in the detection and classification of attacks, **XGBoost** stands out as the best candidate for this task, due to its balance between precision, recall, and F1-score.

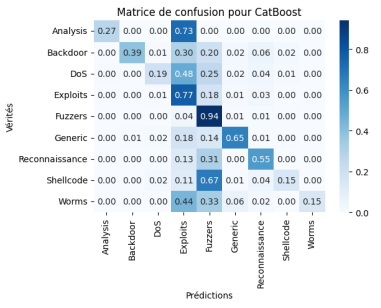


Figure 23: CatBoost

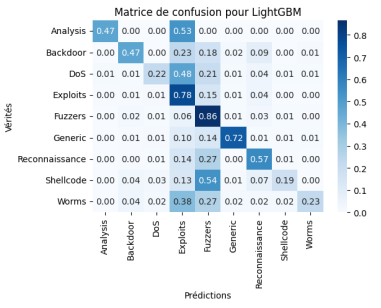


Figure 23: LightGBM

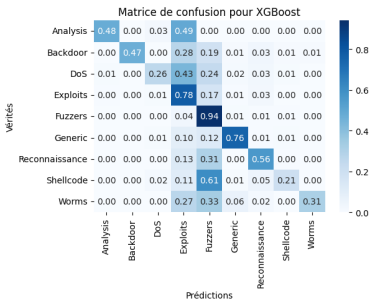


Figure 24: Confusion matrices for CatBoost, LightGBM, and XGBoost in attack classification.

Furthermore, its robustness on majority classes and its relative effectiveness on rare classes reinforce its role as a preferred tool in multi-class attack detection.

To further enhance the performance of the **XGBoost** model in multi-class classification, a search for optimal hyperparameters was conducted using the **Optuna** library. The results after optimization are detailed below:

- **Overall Precision** : 0.7534
- **Overall F1-Score (weighted average)** : 0.74
- **Class "Analysis"** :
  - Precision : 0.55
  - Recall : 0.48
  - F1-Score : 0.51
- **Class "Backdoor"** :
  - Precision : 0.96
  - Recall : 0.49
  - F1-Score : 0.65
- **Class "DoS"** :
  - Precision : 0.75
  - Recall : 0.28
  - F1-Score : 0.41
- **Class "Exploits"** :
  - Precision : 0.82
  - Recall : 0.78
  - F1-Score : 0.80
- **Class "Fuzzers"** :
  - Precision : 0.68
  - Recall : 0.94
  - F1-Score : 0.79

- **Class "Generic"** :
  - Precision : 0.87
  - Recall : 0.78
  - F1-Score : 0.82
- **Class "Reconnaissance"** :
  - Precision : 0.84
  - Recall : 0.55
  - F1-Score : 0.67
- **Class "Shellcode"** :
  - Precision : 0.48
  - Recall : 0.19
  - F1-Score : 0.28
- **Class "Worms"** :
  - Precision : 0.74
  - Recall : 0.35
  - F1-Score : 0.48

Comparing these results with those obtained before optimization, it is observed that :

- Overall precision has increased (from 0.7522 to 0.7534), indicating a slight improvement in overall performance.
- Some classes, such as **Exploits**, **Fuzzers**, and **Generic**, maintain high recall and F1-Score, confirming their robustness after optimization.
- For minority classes such as **Backdoor** and **Worms**, optimization led to a notable improvement in precision and recall.
- Classes such as **Shellcode** and **DoS** still present margins for improvement, particularly in recall and F1-Score.

These results confirm that hyperparameter optimization slightly improves overall performance and encourages a better balance between precision and recall for some critical classes. However, specific improvements are still needed to better address less frequent classes.

In conclusion, the use of Optuna to adjust the hyperparameters of **XGBoost** has led to optimal results in multi-class classification while maintaining strong performance for majority classes. These adjustments make **XGBoost** a high-performing and suitable model for this complex task.

3.2.3. Attack Identification (Ensemble Models)

To go beyond the performance achieved with individual models (see previous sections), we evaluated different ensemble approaches for binary attack detection (*Identification*). Three complementary strategies were considered:

- **Voting**, which combines the predictions of several classifiers to make a majority decision or one based on the sum of probabilities.
- **Bagging**, where several instances of the same algorithm (here `XGBoost`) are trained on randomly sampled (with replacement) subsets of the dataset, and then aggregated.
- **Stacking**, which nests several base models (here `XGBoost`, `LightGBM`, and `CatBoost`) and utilizes a meta-model (`XGBoost`) to learn from their predictions.

Evaluation Metrics: In the context of attack detection, we primarily relied on four metrics to evaluate and compare the performance of the models:

- **Accuracy**: Proportion of correct predictions (well-detected attacks + correctly identified benign flows) over all predictions. Accuracy can be misleading in cases of significant class imbalance, but it remains a general performance indicator.
- **Precision**: Fraction of positive predictions (detected attacks) that are actually positive. More concretely, among all flows declared malicious by the model, Precision indicates the proportion that is indeed malicious. It is crucial for reducing false positives.
- **Recall**: Fraction of positive instances (true attacks) that are correctly detected. It measures the model's ability to recover actual malicious elements. A high Recall is essential to minimize false negatives.
- **F1-Score**: Harmonic mean between Precision and Recall. The F1-Score is particularly useful for comparing models when there is class imbalance (malicious vs benign), as it balances the importance given to false positives and false negatives.

Ensemble Models Without Hyperparameter Optimization: Initially, these ensemble algorithms were evaluated without specific hyperparameter tuning for each classifier. The results obtained (Accuracy, Precision, Recall, and F1-Score) are indicated below:

- **Voting Classifier**:
  - *Accuracy* ≈ 0.97769
  - *Precision* ≈ 0.92872
  - *Recall* ≈ 0.99199
  - *F1-Score* ≈ 0.95931
- **Bagging (XGBoost)**:
  - *Accuracy* ≈ 0.97748
  - *Precision* ≈ 0.92862
  - *Recall* ≈ 0.99125
  - *F1-Score* ≈ 0.95891
- **Stacking (XGBoost, LightGBM, CatBoost)**:
  - *Accuracy* ≈ 0.97751
  - *Precision* ≈ 0.92967
  - *Recall* ≈ 0.99008
  - *F1-Score* ≈ 0.95892

We observe that **Voting** stands out slightly in *Accuracy* and *Recall*, while **Stacking** offers the best *Precision*. Nevertheless, the performance gaps remain relatively small, with each approach showing notable robustness for binary attack detection.

Ensemble Models with Hyperparameter Optimization: To further improve performance, automated optimization strategies (via `Optuna`) were applied to each base algorithm (`XGBoost`, `LightGBM`, `CatBoost`), and these optimized classifiers were then used in ensemble methods. The final results for binary detection are summarized below:

- **Voting Classifier (after optimization)**:
  - *Accuracy* ≈ 0.97738
  - *Precision* ≈ 0.92692
  - *Recall* ≈ 0.99298
  - *F1-Score* ≈ 0.95881
- **Bagging (optimized XGBoost)**:
  - *Accuracy* ≈ 0.97724
  - *Precision* ≈ 0.92757
  - *Recall* ≈ 0.99156
  - *F1-Score* ≈ 0.95850
- **Stacking (optimized XGBoost, LightGBM, CatBoost)**:
  - *Accuracy* ≈ 0.97750
  - *Precision* ≈ 0.92882
  - *Recall* ≈ 0.99107
  - *F1-Score* ≈ 0.95893

In this configuration, **Stacking** achieves a slight advantage in *Accuracy* (≈ 0.97750) and *F1-Score* (≈ 0.95893), while **Voting** shows a slightly higher *Recall* (≈ 0.99298), which is critical for minimizing false negatives. The improvements confirm the value of fine-tuning hyperparameters, although the gains remain relatively modest.

Conclusion (Binary Identification): The application of ensemble methods (Voting, Bagging, Stacking) has consolidated and refined the already high performance of the best individual algorithms for attack detection. Depending on the primary objective (*reduce false positives* or *reduce false negatives*), the most appropriate approach will be favored:

- **Stacking** for its balance of *Precision*/*Recall* and high *F1-Score*,
- **Voting** when *Recall* is critical (monitoring aimed at capturing nearly all threats).

Thus, ensemble approaches fully validate their contribution to the rapid and reliable detection of malicious flows.

3.2.4. Attack Classification (Ensemble Models)

To go beyond the performances obtained with individual models (see previous sections), we evaluated different ensemble approaches for **attack classification**. Three complementary strategies were considered:

- **Voting**, which combines the predictions of several classifiers to make a majority decision or based on the sum of probabilities.
- **Bagging**, where multiple instances of the same algorithm (here `XGBoost`) are trained on randomly drawn (with replacement) samples from the dataset, then aggregated.
- **Stacking**, which nests several base models (for example `XGBoost`, `LightGBM`, and `CatBoost`) and uses a meta-model to learn from their predictions.

Evaluation Measures: In the context of multiclass attack classification, we mainly relied on four metrics to evaluate and compare the models' performances:

- **Accuracy**: Proportion of correct predictions (well-classified attacks + correctly identified benign traffic) over all predictions.
- **Precision**: Fraction of positive predictions in each class that are actually positive.
- **Recall**: Fraction of positive instances (true attacks) in each class that are correctly detected.
- **F1-Score**: Harmonic mean between Precision and Recall, useful for balancing the importance of false positives and false negatives when classes are imbalanced.

Ensemble Models Without Hyperparameter Optimization: Initially, these ensemble algorithms were evaluated **without specific adjustment** of hyperparameters for each classifier. The obtained results (Accuracy, Precision, Recall, and F1-Score) are indicated below:

- **Voting Classifier**:
  - *Accuracy* ≈ 0.74042
  - *Precision* ≈ 0.74166
  - *Recall* ≈ 0.74042
  - *F1-Score* ≈ 0.72510
- **Bagging (XGBoost)**:
  - *Accuracy* ≈ 0.74775
  - *Precision* ≈ 0.76145
  - *Recall* ≈ 0.74775
  - *F1-Score* ≈ 0.73130
- **Stacking**:
  - *Accuracy* ≈ 0.74769
  - *Precision* ≈ 0.75468
  - *Recall* ≈ 0.74769
  - *F1-Score* ≈ 0.73565

Analysis (Without optimization):
The **Voting Classifier** shows the lowest accuracy (0.74042) and a lower F1-Score (0.72510) than the other methods. **Bagging** and **Stacking** demonstrate very close performances (Accuracy around 0.7477), with a slight advantage in Precision for Bagging (0.76145) and a slightly better F1-Score for Stacking (0.73565). Overall, **Bagging and Stacking slightly outperform** Voting, though without creating a significant gap between them.

Ensemble Models with Hyperparameter Optimization: To further enhance performance, optimization strategies (for example via `Optuna`) were applied to the base algorithms, and these optimized classifiers were then used in the ensemble methods. The final results for attack classification are summarized below:

- **Voting Classifier (after optimization)**:
  - *Accuracy* ≈ 0.75397
  - *Precision* ≈ 0.76952
  - *Recall* ≈ 0.75397
  - *F1-Score* ≈ 0.73844
- **Bagging (optimized XGBoost)**:
  - *Accuracy* ≈ 0.74898
  - *Precision* ≈ 0.76253
  - *Recall* ≈ 0.74898
  - *F1-Score* ≈ 0.73329
- **Stacking (optimized)**:
  - *Accuracy* ≈ 0.75650
  - *Precision* ≈ 0.76727
  - *Recall* ≈ 0.75650
  - *F1-Score* ≈ 0.74464

Analysis (With optimization):
All three approaches **benefit significantly from hyperparameter optimization**, as illustrated by their respective increases in Accuracy and F1-Score. The **Voting Classifier**, for example, moves from 0.74042 to 0.75397 in Accuracy, indicating a substantial gain. The **Voting** achieves the *highest absolute Precision* (0.76952), while **Stacking** leads in Accuracy (0.75650) and F1-Score (0.74464). The **optimized Bagging** ranks in between in terms of overall performance (Accuracy at 0.74898, F1-Score at 0.73329), but remains below Stacking. Ultimately, the **optimized Stacking** emerges as the most balanced method, offering the best Precision/Recall combination and the highest F1-Score.

Conclusion (Attack Classification): **Hyperparameter optimization** brings considerable gains for all three ensemble methods, confirming the usefulness of fine-tuning to improve multiclass attack detection. If the goal is to *maximize* precision (reduce false positives), one might lean towards the Voting Classifier, while the **optimized Stacking**, due to its balance between Precision and Recall, **offers the best overall performance** for attack classification.

Table 1
Comparison of different results for Identification and Classification (with or without optimization).

Type	Hyperparameters	Technique	Accuracy	Precision	Recall	F1 Score
Identification	False	Individual	0.977497	0.962880	0.981657	0.971709
Identification	False	Voting	0.977692	0.928720	0.991992	0.959314
Identification	False	Bagging	0.977480	0.928617	0.991253	0.958913
Identification	False	Stacking	0.977513	0.929666	0.990083	0.958924
Identification	True	Individual	0.977317	0.927489	0.991992	0.958657
Identification	True	Voting	0.977382	0.926916	0.992978	0.958810
Identification	True	Bagging	0.977235	0.927567	0.991561	0.958497
Identification	True	Stacking	0.977497	0.928819	0.991068	0.958934
Classification	False	Individual	0.752187	0.725799	0.528686	0.586843
Classification	False	Voting	0.740421	0.741662	0.740421	0.725102
Classification	False	Bagging	0.747752	0.761448	0.747752	0.731302
Classification	False	Stacking	0.747690	0.754685	0.747690	0.735648
Classification	True	Individual	0.753357	0.766266	0.753357	0.739967
Classification	True	Voting	0.753973	0.769516	0.753973	0.738445
Classification	True	Bagging	0.748984	0.762530	0.748984	0.733294
Classification	True	Stacking	0.756499	0.767270	0.756499	0.744639

3.2.5. Conclusion of Supervised Machine Learning

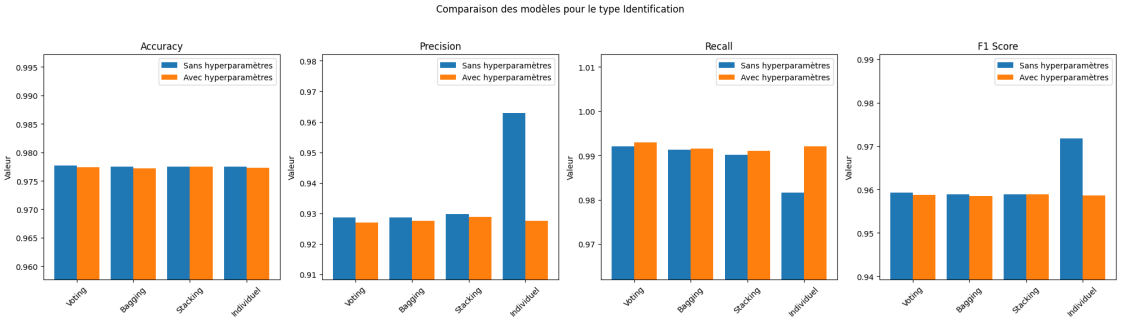


Figure 25: (a) Comparison of models for the Identification task

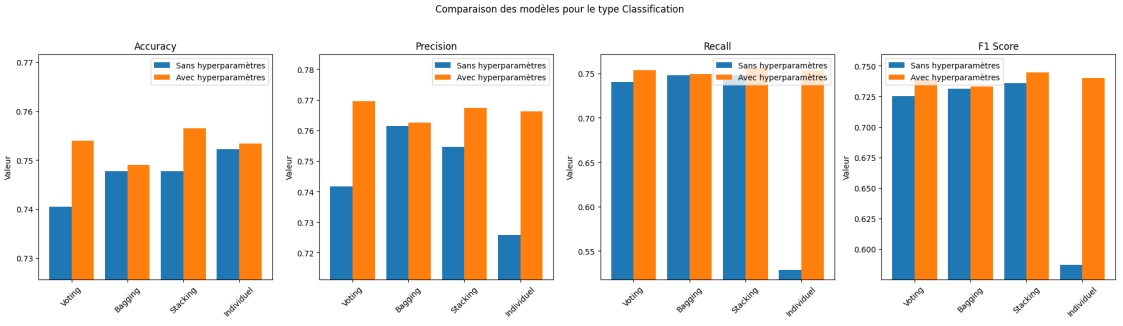


Figure 26: (b) Comparison of models for the Classification task

We can make the following observations based on both table 1 and the graphs in figures 25 and 26 :

Identification (Binary Task). Focusing on the rows dedicated to *Identification* (**14, 0, 1, 2** for the absence of optimization and **12, 6, 7, 8** for optimization), we note:

- **Very high performances:** Accuracy scores are close to 0.977 and the F1 Score hovers around 0.959, for ensemble methods (*Voting*, *Bagging*, *Stacking*) as well as for the *Individual* approach, with or without optimization.
- **Limited gains from optimization:** The differences between "Without" and "With hyperparameters" remain minimal (e.g., *Stacking*: 0.977513 vs 0.977497). The *Voting* (0.977692 vs 0.977382) also shows a negligible gap.
- **Tight visual comparison:** Figure 25 shows very close bars, illustrating that all these models are nearly at the same performance level. *Stacking* stands out slightly in *Precision*, but the difference remains small.
- **Role of the optimized individual model:** Even if it does not always have the highest Accuracy, this model proves to be very competitive (e.g., row 12: 0.977317 Accuracy and 0.958657 F1 Score), emphasizing that good optimization of a single classifier can rival ensemble approaches.

Classification (Multiclass Task). Referring to rows **15, 3, 4, 5** (without optimization) and **13, 9, 10, 11** (with optimization):

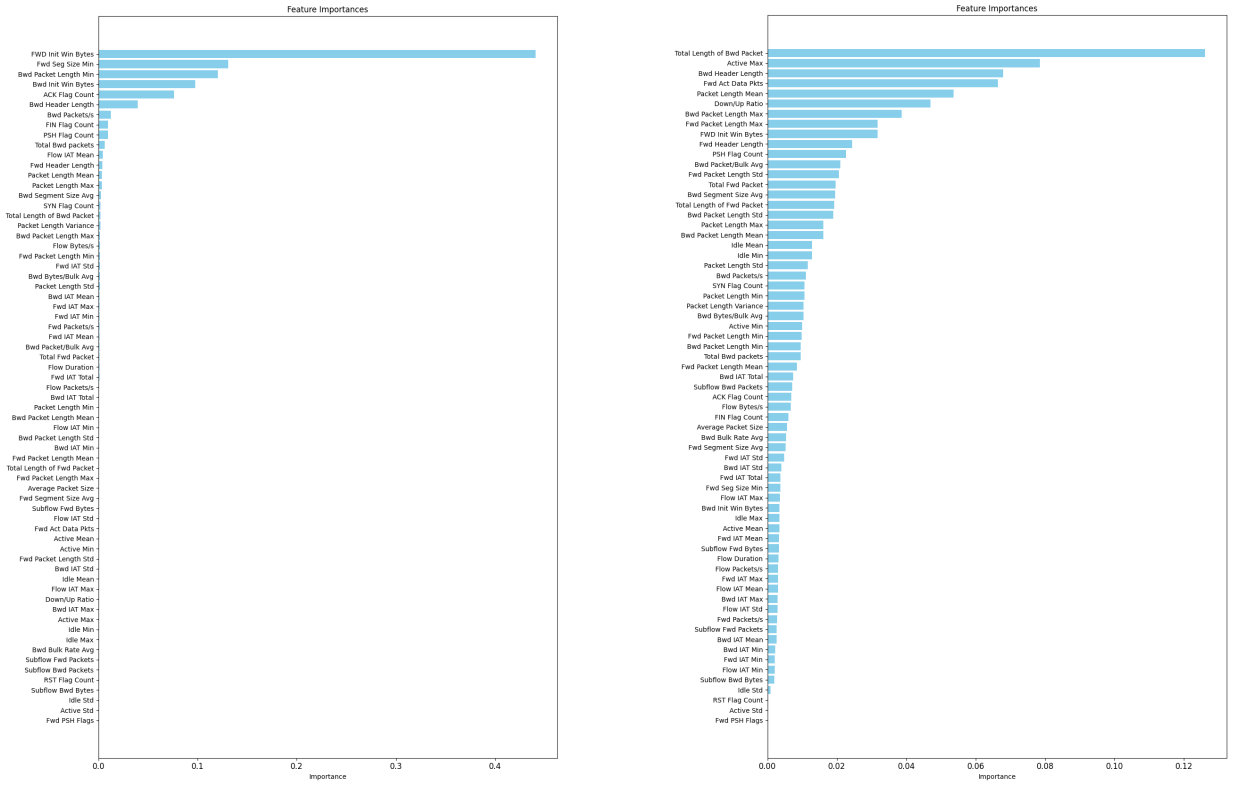
- **More significant impact of optimization:** Notable gains are observed, especially for *Voting* which increases from 0.740421 (row 3) to 0.753973 (row 9).
- **Stacking optimizes even better:** With 0.756499 in Accuracy and 0.744639 in F1 Score (row 11), the *optimized Stacking* slightly dominates its competitors. Visually, figure 26 shows it a notch above *Voting* and *Bagging*.
- **Bagging and Voting close:** Although they improve significantly due to optimization, their performances (row 9 and row 10) remain slightly below Stacking at the F1 Score level. Nevertheless, *Voting* retains the highest Precision (0.769516).
- **Individual Models:** The individual approach (row 13) still achieves 0.753357 in Accuracy, a level similar to the non-optimized ensemble methods, proving that *optimization can, in some cases, compensate for the lack of aggregation techniques*.

In conclusion, **all models provide a very solid level of performance for detecting and classifying attacks**, but the choice of the method will depend on the targeted objective:

- **Minimize false negatives (exhaustive detection):** The results show that *Voting* and *Stacking* consistently maintain a high *Recall*, especially in binary mode (*Identification*). These approaches are therefore particularly suited when one wants to detect as many attacks as possible, even if this comes with a slight risk of additional false positives.
- **Reduce false positives (operational stability):** If the primary goal is to avoid excessive alerts, *Bagging* and *Voting* often show competitive *Precision* (e.g., ≈ 0.769516 for Voting in *Classification*). They therefore allow for better control of the number of legitimate flows incorrectly identified as malicious.
- **Achieve the best overall balance:** The *optimized Stacking* takes a *slight lead* over other methods in several scenarios (e.g., Accuracy and F1 Score in *Classification*), making it more versatile when one wishes to maintain an optimal compromise between false positives and false negatives.
- **Computational budget or practical constraints:** An *optimized individual model* (e.g., XGBoost alone) can remain relevant if one seeks a simpler solution to deploy while maintaining performance close to that of ensemble methods.

3.2.6. Feature Importances

Feature Importances for Individual Adjusted Models Figure 27 shows the feature importances for the individual models adjusted respectively for the binary identification task and the multi-class classification task. These analyses highlight the most influential features used by the models to make their predictions.

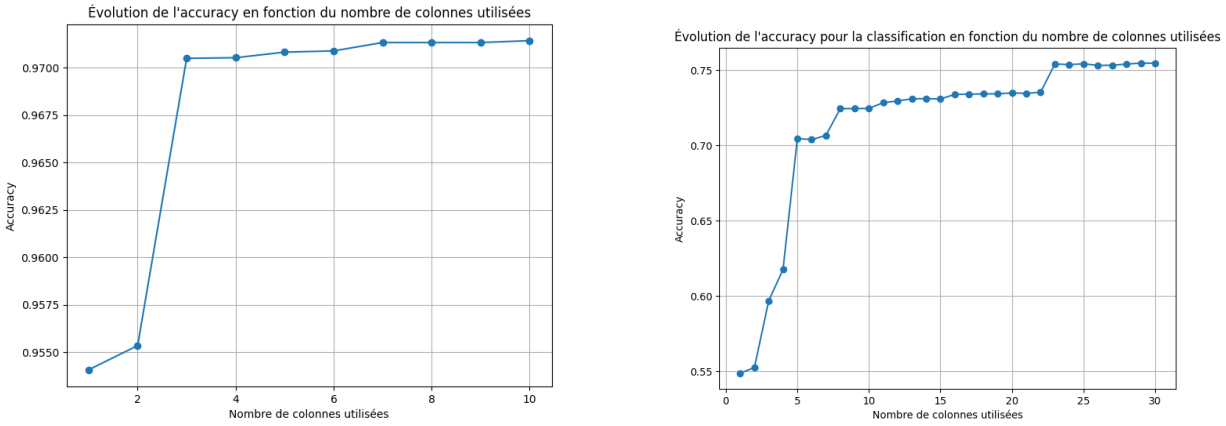


(a) **Binary Identification** (b) **Multi-Class Classification**

Figure 27: Feature importances for the binary identification and multi-class classification tasks.

Analysis: For the identification task, the most important features include `FWD Init Win Bytes`, `Fwd Seg Size Min`, and `Bwd Packet Length Min`. These features are essential for distinguishing benign flows from malicious flows. For the multi-class classification, the most important features include `Total Length of Bwd Packet`, `Active Max`, and `Bwd Header Length`, which efficiently distinguish different categories of attacks.

Impact on Accuracy Based on the Number of Features Figure 28 illustrates the evolution of accuracy based on the number of features used for identification and classification tasks.



(a) **Binary Identification** (b) **Multi-class Classification**

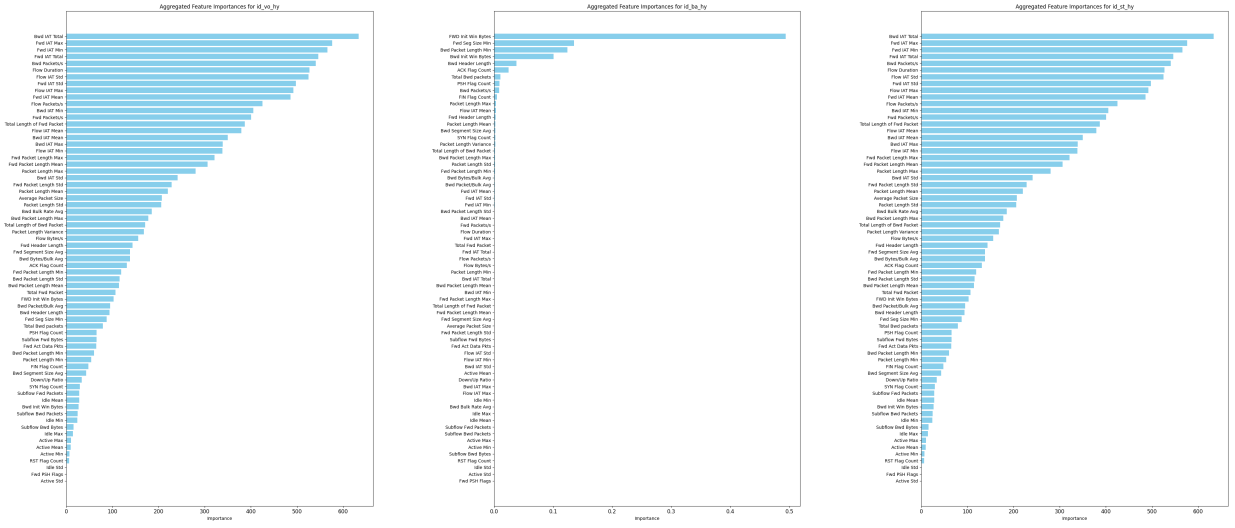
Figure 28: Evolution of accuracy based on the number of features for identification and classification tasks.

Analysis: For binary identification, a rapid improvement in accuracy is observed with the addition of the first features, reaching a plateau around 4 features. For multi-class classification, accuracy continues to improve with the addition of new features, although the gains become marginal beyond 15 to 20 features. This reflects the increased complexity of the classification task.

Conclusion These analyses demonstrate the importance of features specific to network flows (e.g., packet sizes, durations, connection parameters) in model performance. The optimal choice of features may vary depending on the task (binary or multi-class), and a judicious selection helps to maximize performance while reducing complexity.

Feature Importances for Adjusted Ensemble Models

Identification (Ensemble Models) In this section, we analyze the feature importances of adjusted ensemble models for identification. Three main models were used: Voting, Bagging, and Stacking. Each model combines the predictions of several sub-models to enhance robustness and accuracy. The graphs below illustrate the feature importances for each model, presented side by side to allow for visual comparison.



(a) **Voting Model** (b) **Bagging Model** (c) **Stacking Model**

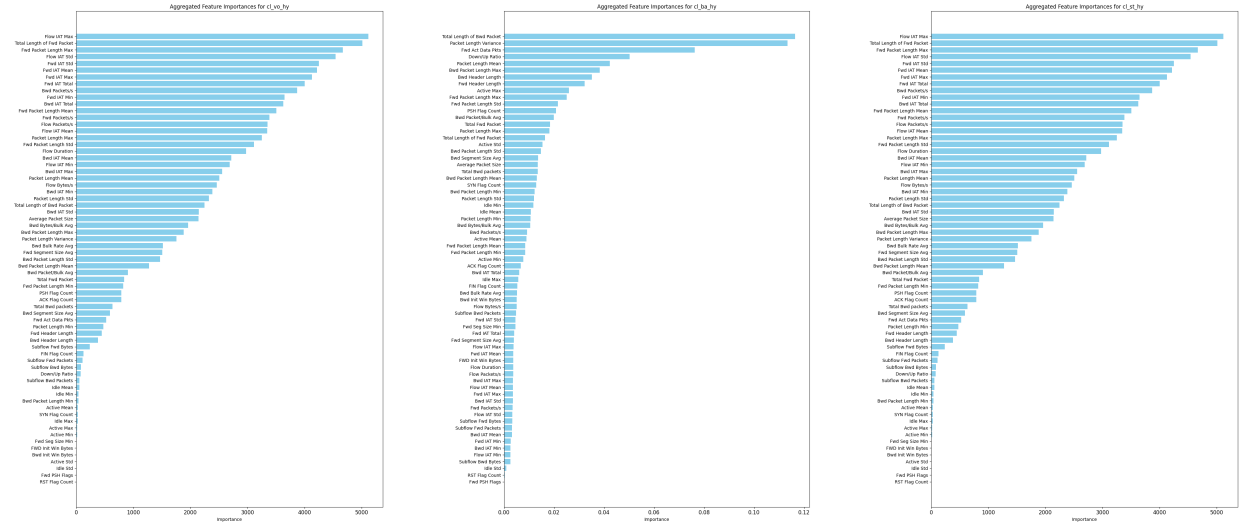
Figure 29: Feature importances for adjusted ensemble models (Voting, Bagging, and Stacking).

The results show that some models share important features, while others have distinct specifics:

- All three models identify *FWD Init Win Bytes* as a key feature. This reflects its role in the initial configuration of network flows.
- The Bagging and Stacking models prioritize temporal features such as *Bwd IAT Total* and *Fwd IAT Mean*, which allow for the detection of temporal patterns and variations in traffic.
- The Voting model focuses more on packet-related attributes like *Fwd Seg Size Min* and *ACK Flag Count*, suggesting a protocol-oriented approach.

These observations highlight the importance of temporal features for complex models like Bagging and Stacking, while static features are sufficient for simpler models like Voting. This underscores the complementarity of approaches and the necessity of choosing a model suited to the nature of the data and the problem being studied.

Classification (Ensemble Models) In this section, we analyze the feature importances of ensemble models fitted for multi-class classification. Three main models were used: Voting, Bagging, and Stacking. The graphs below illustrate the feature importances for each model, presented side by side to facilitate visual comparison.



(a) **Voting Model** (b) **Bagging Model** (c) **Stacking Model**

Figure 30: Feature importances for the fitted ensemble models (Voting, Bagging, and Stacking) in multi-class classification.

The results show that each model assigns variable importance to the features used in multi-class classification. Here are the main observations:

- **Voting Model:** The most important features include *Flow IAT Max*, *Total Length of Fwd Packet*, and *Fwd Packet Length Max*. These results indicate that this model emphasizes temporal metrics and packet sizes to differentiate classes. The consistency of these features shows their relevance for this type of aggregation.
- **Bagging Model:** The Bagging model favors features such as *Total Length of Bwd Packet*, *Packet Length Variance*, and *Fwd Act Data Pkts*. These results suggest that this model is more sensitive to variations in length and the volume of transferred data, reflecting its ability to reduce variance in the sub-models.
- **Stacking Model:** For Stacking, the dominant features include *Flow IAT Max*, *Total Length of Fwd Packet*, and *Fwd Packet Length Std*. This indicates that the Stacking model uses similar temporal metrics as the Voting Model but incorporates a distribution aspect (such as the standard deviation of packet lengths).
- **Common Features:** Several features are important for all three models, notably *Flow IAT Max* and *Total Length of Fwd Packet*. This suggests that these metrics are generally relevant for multi-class classification, regardless of the ensemble approach used.

These observations highlight the complementarity of different ensemble models. While Voting and Stacking emphasize similar temporal metrics, Bagging further explores the variations in lengths and transferred data. These differences enhance our understanding of how each model utilizes the data to perform robust classifications.

3.2.7. SHAP

SHAP for Adjusted Individual Models In this section, we explore the SHAP (SHapley Additive exPlanations) values and interactions for adjusted individual models, focusing on identification and classification tasks.

Identification (Individual Model) The graphs below illustrate the individual contributions of features (SHAP values) for the identification model.

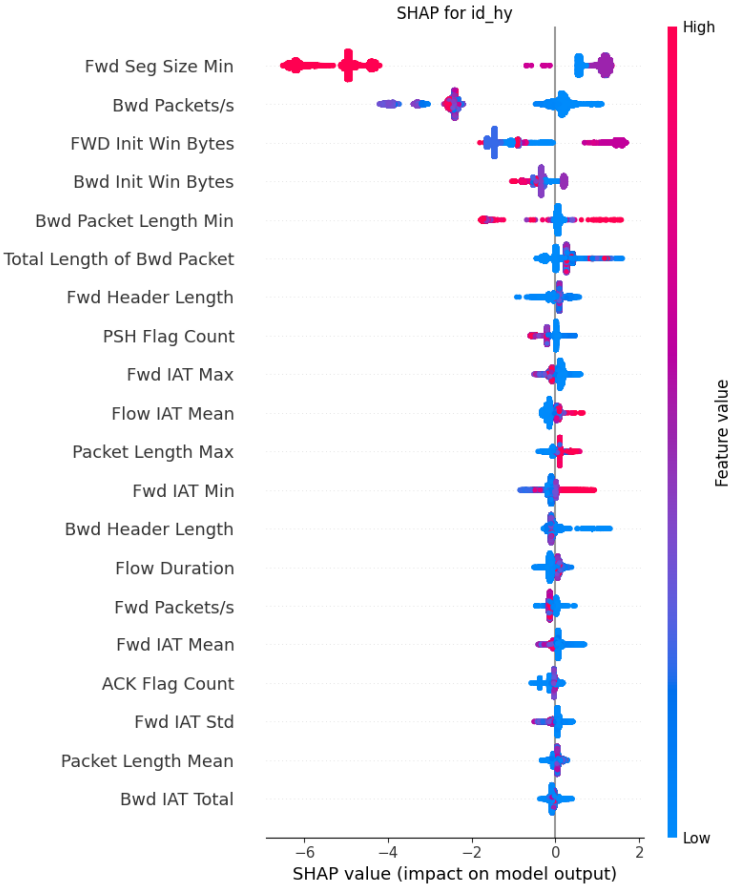


Figure 31: SHAP values for the individual identification model.

The results show that the feature *Fwd Seg Size Min* has a significant and direct contribution to the prediction. High values of this feature are associated with a positive impact on the model. Other features such as *Bwd Packets/s* and *FWD Init Win Bytes* also play a key role in the identification model.

Classification (Individual Model) For individual models dedicated to classification, the graphs below show the SHAP interactions between the different features.

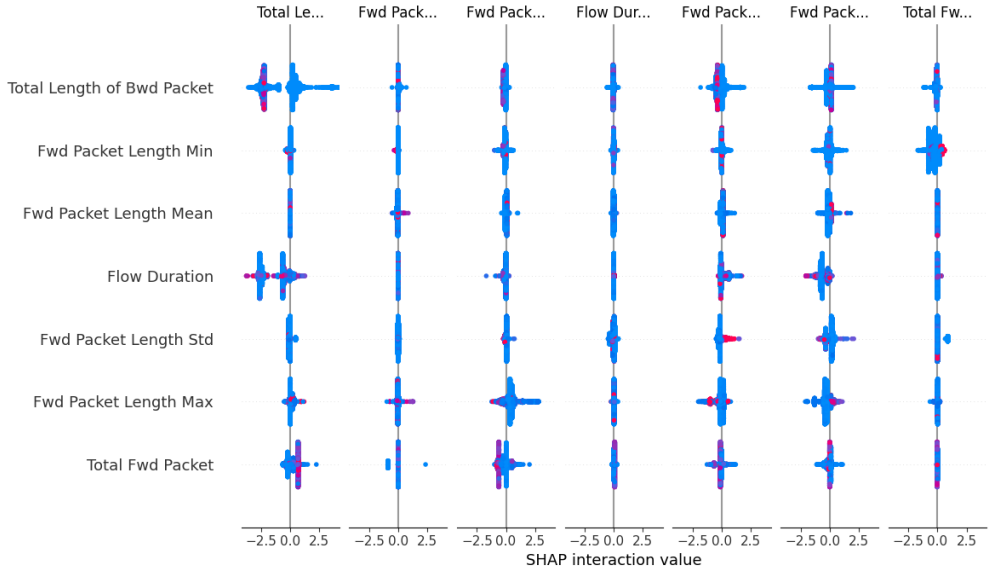


Figure 32: SHAP interactions for the individual classification model.

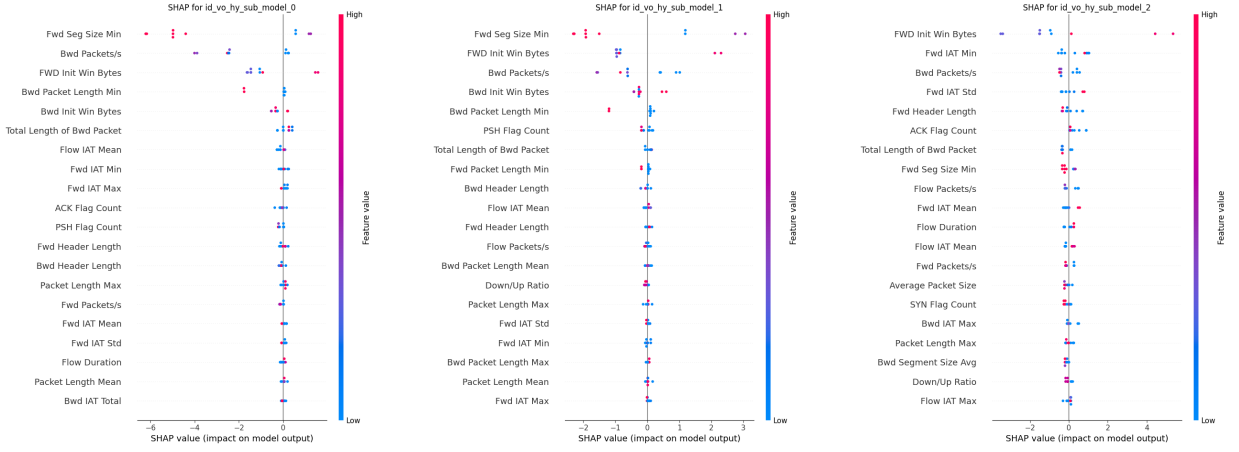
In the context of classification, the interactions reveal complex combined effects between key features such as *Total Length of Bwd Packet*, *Fwd Packet Length Min*, and *Flow Duration*. These interactions emphasize the importance of analyzing dependencies between features to enhance the understanding of the model's predictions.

SHAP for Adjusted Ensemble Models

Identification (Ensemble Models) In this section, we analyze the SHAP values for adjusted ensemble models within the framework of the identification task. The three models studied are Voting, Bagging, and Stacking. The associated graphs illustrate the individual contributions of the features for each sub-model.

Voting Model

The graphs below present the SHAP values for the sub-models of the Voting model.



(a) **Sub-model 1** (b) **Sub-model 2** (c) **Sub-model 3**

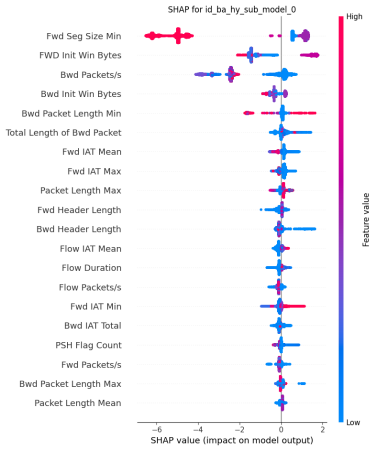
Figure 33: SHAP values for the sub-models of the Voting model.

The results of the three graphs show that certain features, such as **Fwd Seg Size Min** and **Bwd Packets/s**, are frequently identified as having a strong contribution to the predictions. In the case of **Sub-model 1**, **Fwd Seg Size Min** plays a predominant role, indicating a strong correlation with the model's decisions. For **Sub-model 2**, importance is also attributed to **FWD Init Win Bytes** and **Bwd Packet Length Min**, suggesting that these features significantly influence the predictions. Finally, in **Sub-model 3**, **Flow Duration** and **Total Length of Bwd Packet** stand out, highlighting variations in the priorities of the sub-models.

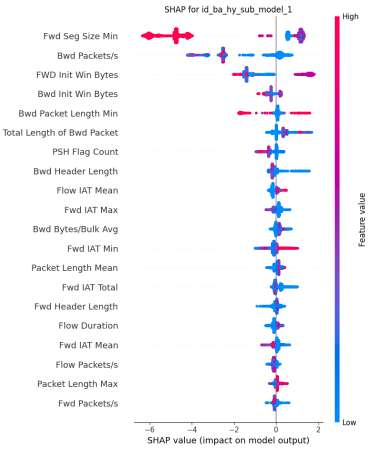
These observations underscore the diversity of information used by the Voting sub-models, while also emphasizing certain recurring features that serve as pillars for the predictions. This illustrates the complementarity of the sub-models within this ensemble framework.

Bagging Model

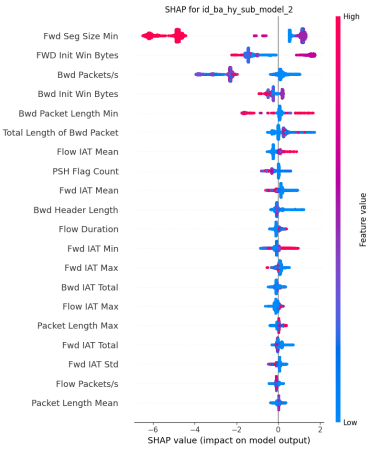
The following graphs group the SHAP values for the five sub-models of the Bagging model.



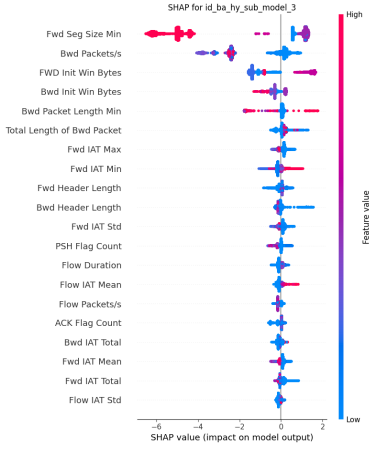
(a) **Sub-model 1**



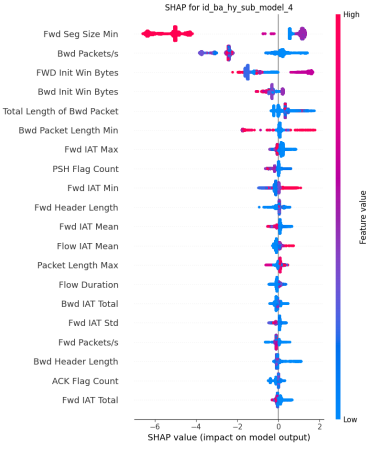
(b) **Sub-model 2**



(c) **Sub-model 3**



(d) **Sub-model 4**



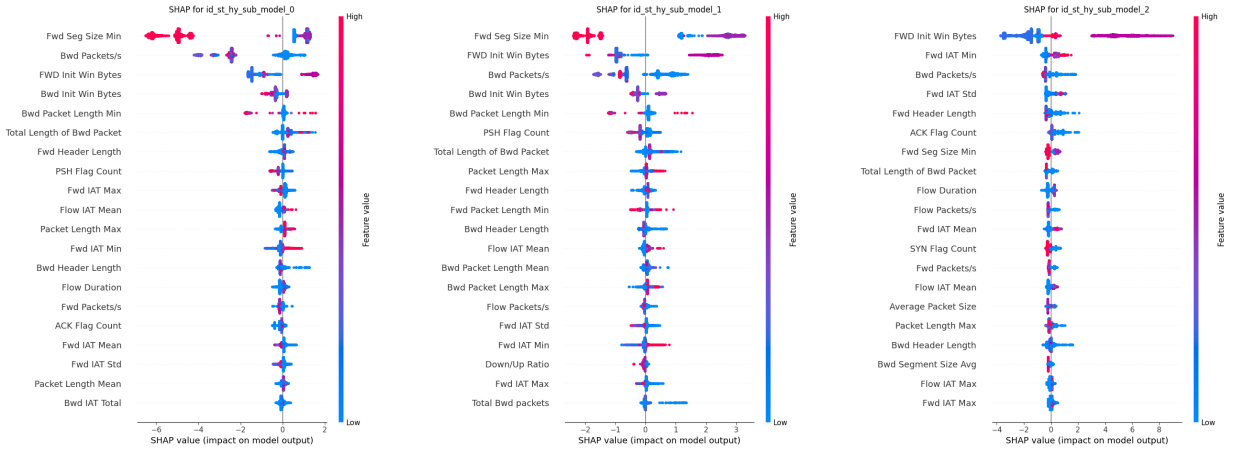
(e) **Sub-model 5**

Figure 34: SHAP values for the sub-models of the Bagging model.

The analyses reveal that features such as **Fwd Seg Size Min** and **Bwd Packets/s** are consistently important across the five sub-models. However, certain features, such as **Total Length of Bwd Packet** and **Packet Length Variance**, show notable variations in their impact depending on the particular sub-model considered. These differences can be attributed to the specifics of each sub-model within the Bagging framework, where random samples of the data influence the importance prioritization of features.

Stacking Model

Finally, the graphs below present the SHAP values for the three sub-models of the Stacking model.



(a) **Sub-model 1** (b) **Sub-model 2** (c) **Sub-model 3**

Figure 35: SHAP values for the sub-models of the Stacking model.

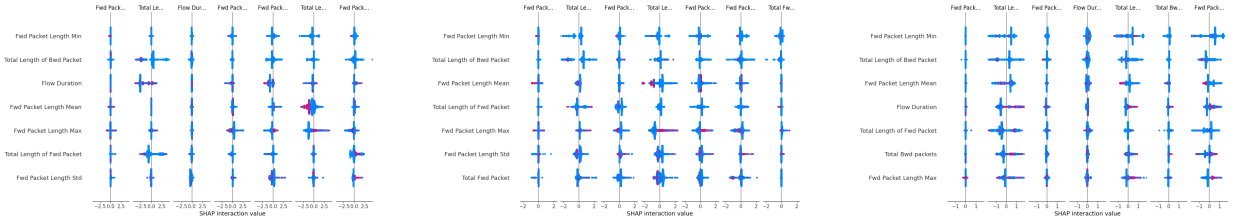
The results of the three sub-models of the Stacking model indicate a strong influence of **Fwd Seg Size Min** and **Bwd Packets/s** on the predictions. These features are recurrent and show consistent importance across the sub-models, although the specific contributions vary slightly depending on the sub-model.

In addition, features such as **Fwd IAT Mean** and **Flow Duration** also provide significant contributions, but their impact seems to vary more depending on the sub-model. The overall analysis highlights that the Stacking model leverages a wide range of features to make robust predictions while maintaining some consistency in the importance of key features.

Classification (Ensemble Models) In this section, we analyze the SHAP values for the ensemble models fitted for the classification task. The three models studied are Voting, Bagging, and Stacking. The associated graphs illustrate the individual contributions of features for each sub-model.

Voting Model

The graphs below present the SHAP values for the sub-models of the Voting model.



(a) **Sub-model 1** (b) **Sub-model 2** (c) **Sub-model 3**

Figure 36: SHAP values for the sub-models of the Voting model in the classification task.

The results from the graphs for the Voting model indicate that features such as **Fwd Seg Size Min** and **Bwd Packets/s** are consistently identified as having a strong contribution to the predictions.

In **Sub-model 1**, **Fwd Seg Size Min** appears as one of the most influential features, suggesting that the minimum forward segment sizes are critical for classification. **Bwd Packets/s**, representing the density of backwards packets, also plays a key role, indicating that outgoing traffic is an important measure.

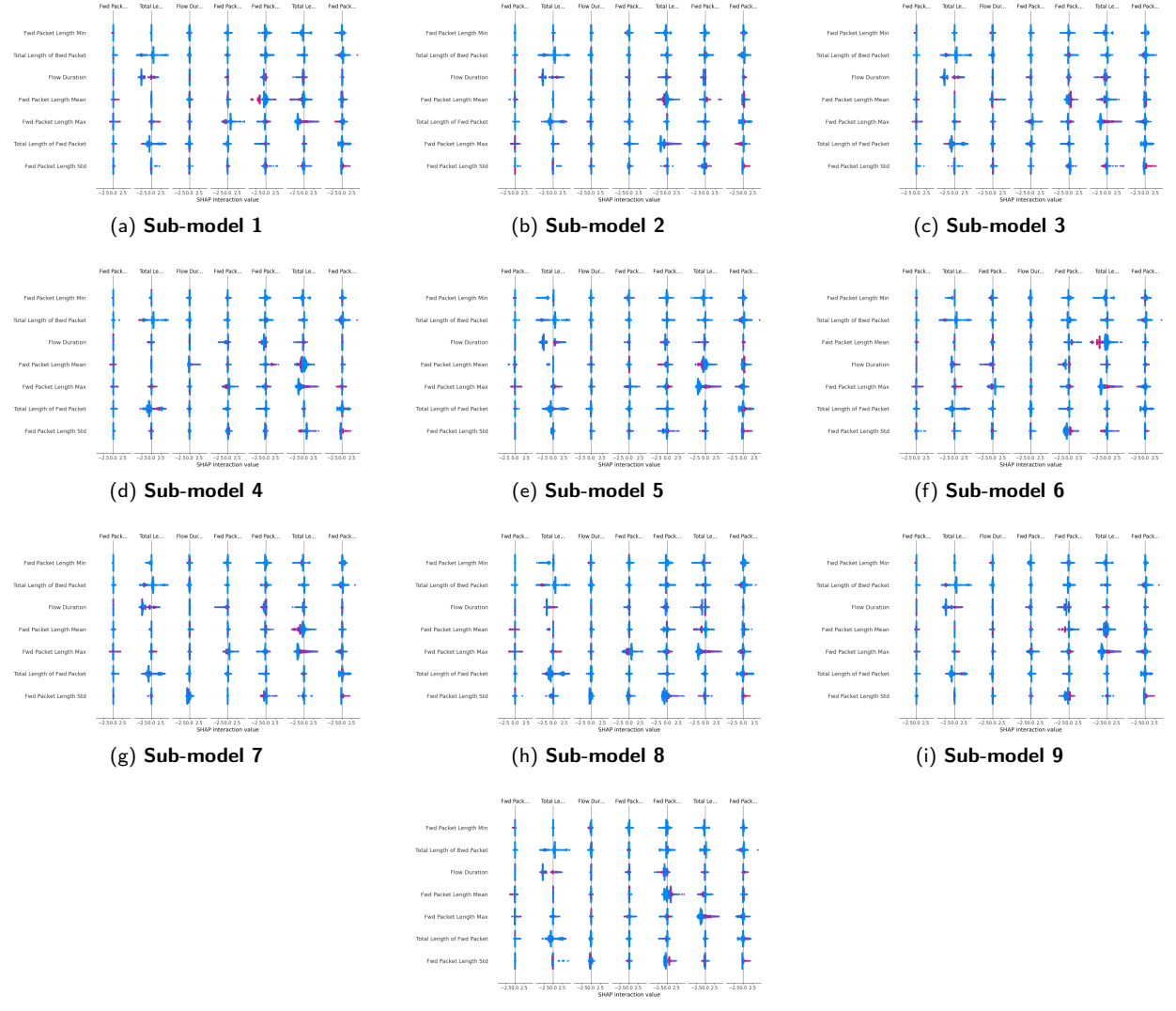
Sub-model 2 highlights additional features like **Fwd IAT Mean** and **Bwd Packet Length Min**, suggesting that the temporal variability of the transmitted segments (IAT) as well as the minimum length of backwards packets influence the model's decisions. This shows the sub-model's ability to capture various behaviors in the data.

Finally, **Sub-model 3** places more emphasis on **Flow Duration** and **Total Length of Bwd Packet**, indicating that the duration of flows and the sum of the lengths of backwards packets have a notable impact in this context.

These variations between the sub-models reflect a complementarity in their operation. Although some attributes are recurrent (such as **Fwd Seg Size Min**), the sub-models also leverage specific features to optimize their predictions. This approach allows the Voting model to combine different perspectives from the data, thereby increasing its robustness in classification.

Bagging Model

The following graphs group the SHAP values for the ten sub-models of the Bagging model.



(a) **Sub-model 1** (b) **Sub-model 2** (c) **Sub-model 3**

(d) **Sub-model 4** (e) **Sub-model 5** (f) **Sub-model 6**

(g) **Sub-model 7** (h) **Sub-model 8** (i) **Sub-model 9**

(j) **Sub-model 10**

Figure 37: SHAP values for the sub-models of the Bagging model in the classification task.

The analysis of the graphs for the Bagging model reveals several interesting observations regarding the contributions of the features used by the sub-models.

First, certain features, such as **Total Length of Bwd Packet** and **Fwd Packet Length Min**, appear recurrently among the most influential. These features play a crucial role in several sub-models, indicating that they capture critical information for the classification task. For example, **Total Length of Bwd Packet** is often associated with significant variations in SHAP values, suggesting that it strongly influences the predictions of the sub-models.

Next, it is noted that the order of importance varies significantly between sub-models. For instance, in some sub-models, **Flow Duration** and **Fwd Packet Length Max** show a high contribution, while in others, these features have a marginal impact. This can be attributed to the Bagging method, which relies on bootstrap samples. Each sub-model is trained on a different subset of the data, which can lead to variations in the selection of important features.

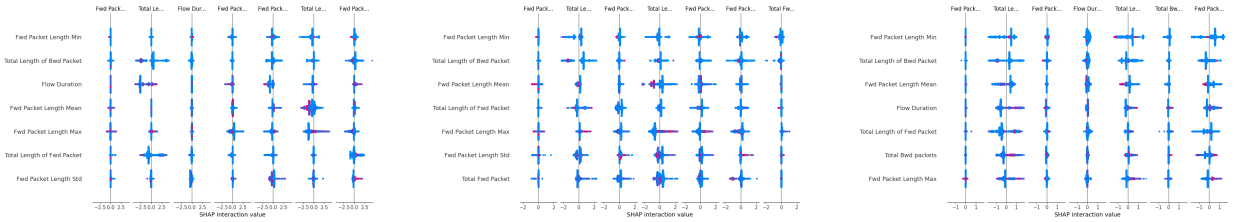
Another notable point is the importance of interactions between features. In several graphs, features such as **Fwd Packet Length Std** and **Packet Length Variance** have a non-linear combined effect on predictions. This highlights that the Bagging model exploits complex relationships between variables to enhance its overall performance.

Finally, although some features are consistently important, others, such as **Bwd IAT Min** or **Fwd IAT Max**, show a significant contribution only in certain sub-models. This reflects the diversity of perspectives brought by the different sub-models, which is one of the main strengths of Bagging. This diversity contributes to the robustness and generalization capability of the model.

In conclusion, the analysis of SHAP values for the Bagging model highlights the importance of key features, the complex interactions between variables, and the complementarity of the sub-models. These observations demonstrate that the Bagging model can leverage different combinations of features to improve its performance in the classification task.

Stacking Model

Finally, the graphs below present the SHAP values for the three sub-models of the Stacking model.



(a) **Sub-model 1** (b) **Sub-model 2** (c) **Sub-model 3**

Figure 38: SHAP values for the sub-models of the Stacking model in the classification task.

The graphs representing the SHAP values for the three sub-models of the Stacking model in the classification task reveal several relevant observations:

- **Sub-model 1** :
  - The features **Fwd Packet Length Min** and **Total Length of Bwd Packet** stand out due to their significant impact on the predictions.
  - Notable variability in SHAP values is observed for features related to flow duration, such as **Flow Duration**, suggesting that this sub-model exploits temporal aspects for its decision.
  - The interactions between **Fwd Packet Length Max** and **Fwd Packet Length Mean** show a strong correlation, indicating that the model assigns simultaneous importance to these two variables.
- **Sub-model 2** :
  - The features **Fwd Packet Length Max** and **Total Length of Fwd Packet** are predominant.
  - A more homogeneous distribution of SHAP values is visible, reflecting a balanced contribution from the selected features.
  - **Total Bwd Packets** and **Fwd Packet Length Std** are moderately influential, highlighting their complementary role in traffic analysis.
- **Sub-model 3** :
  - The features **Flow Duration** and **Fwd Packet Length Min** continue to play a predominant role, but their impact seems to slightly decrease compared to the other two sub-models.
  - **Fwd Packet Length Std** is further emphasized, suggesting an increased sensitivity of the model to variability in packet lengths.
  - A more pronounced correlation between **Total Length of Fwd Packet** and **Fwd Packet Length Max** is observed, reinforcing their joint importance.

The graphs of the Stacking model highlight the importance of **Fwd Seg Size Min** and **Bwd Packets/s**, while showing that the priorities of the sub-models vary slightly. This illustrates the ability of the Stacking model to effectively integrate different perspectives from the sub-models, leading to more robust predictions.

3.2.8. Conclusion Feature Importances and SHAP

The cross-analysis of Feature Importances and SHAP values has highlighted key trends in the influential features for binary identification and multi-class classification tasks, for both individual models and ensemble models. Here are the main observations:

Binary Identification Task

- Both individual and ensemble models identified `Fwd Seg Size Min`, `Bwd Packets/s`, and `FWD Init Win Bytes` as recurring features, suggesting their central role in discriminating benign flows from malicious flows.
- The Voting and Stacking models particularly leverage time-based metrics (`Flow Duration`, `Fwd IAT Mean`), while the Bagging model places more emphasis on length and variance metrics (`Packet Length Variance`, `Total Length of Bwd Packet`).
- SHAP values confirm the importance of these features by showing their direct positive or negative contribution to predictions, with variations depending on the sub-models and ensemble method used.

Multi-Class Classification Task

- The features `Flow IAT Max`, `Total Length of Fwd Packet`, and `Fwd Packet Length Max` emerge as dominant elements in both individual and ensemble models, reflecting their relevance in differentiating attack classes.
- The Bagging and Stacking models are more robust by incorporating complex interactions between features (`Fwd Packet Length Std`, `Total Bwd Packets`), enhancing their ability to capture the specifics of different classes.
- SHAP values also highlight the complementarity of the models, with each model assigning varied weights to features based on their structure and ensemble approach.

Global Implications

- The analyses of Feature Importances and SHAP values have confirmed the following features as critical for both tasks (identification and classification):
  - `Fwd Seg Size Min`
  - `Bwd Packets/s`
  - `FWD Init Win Bytes`
  - `Flow Duration`
  - `Total Length of Fwd Packet`
  - `Flow IAT Max`
  - `Fwd Packet Length Max`
  - `Packet Length Variance`
  - `Total Length of Bwd Packet`

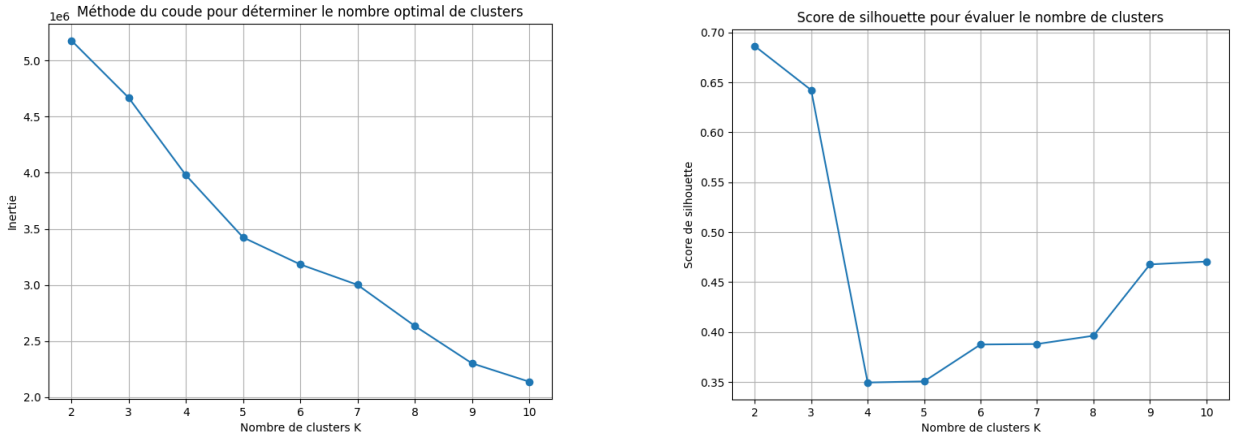
4. Unsupervised Learning

4.1. K-Means

In this section, we present the application of the *K-Means* algorithm for the unsupervised clustering of network flows. The goal is to highlight natural groupings (*clusters*) in the data without using their class labels (`Label Num`).

4.1.1. Elbow Method and Silhouette Score

To determine the optimal number of clusters K, we use the elbow method. In parallel, we evaluate the quality of the cluster separation through the silhouette score. Figure 39 presents both main metrics side by side:



(a) **Elbow Method** (b) **Silhouette Score**

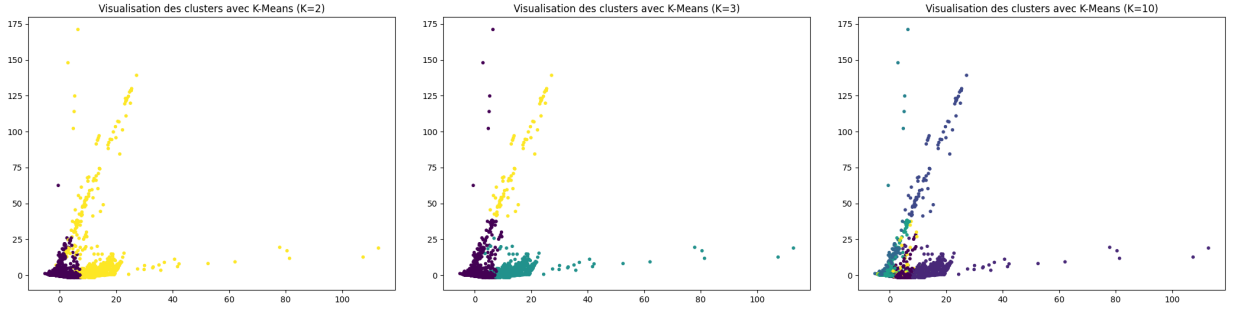
Figure 39: On the left: intra-cluster inertia (elbow method). On the right: silhouette score for different values of K.

The first figure (on the left) illustrates the evolution of intra-cluster inertia (or sum of distances within each cluster) as a function of the number of clusters K. We observe a rapid decrease in inertia up to a certain point where the improvement becomes marginal, forming an elbow around $K \approx 2$ or 3.

The second figure (on the right) presents the silhouette score, which measures intra-cluster cohesion and inter-cluster separation. It ranges from -1 to 1. We see that this score is relatively high for $K = 2$, then decreases beyond this value, reflecting a poorer partition of the data as K increases.

4.1.2. Visualization of Clusters

After analyzing inertia and silhouette score, we retain several values of K to examine the organization of the data through dimensionality reduction (*PCA* with 2 components). Figures 40a, 40b and 40c illustrate the *clustering* for $K = 2$, $K = 3$ and $K = 10$, respectively.



(a) **K = 2** (b) **K = 3** (c) **K = 10**

Figure 40: Visualization of K-Means clusters for different values of K (2D PCA projection).

4.1.3. Analysis and Interpretation

- **K = 2 :** The silhouette score is the highest, suggesting the existence of two well-separated groups. In Figure 40a, we observe two large clouds of points, distinctly separated in the projected space.
- **K = 3 :** A third cluster appears, providing finer segmentation (Figure 40b), but the average silhouette decreases, indicating some additional overlap.
- **K = 10 :** When the number of clusters is pushed to 10 (Figure 40c), we notice a significant fragmentation of the data and a drop in silhouette score, indicating a less relevant partition.

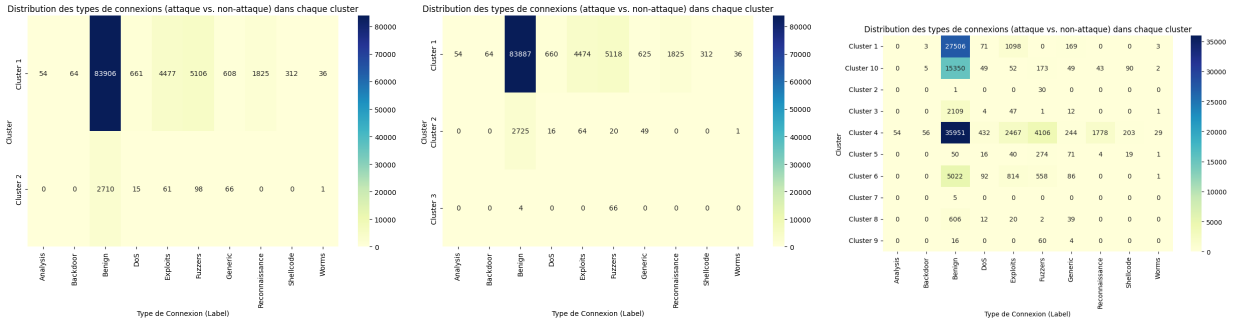
Thus, based on the results of *K-Means*, we have chosen to **keep the values** $K = 2$**,** $K = 3$ **and** $K = 10$ for the continuation of the analysis:

- **K = 2** shows a high silhouette score and may be relevant for **distinguishing two main categories**, for example, detecting attacks in a binary approach (attacks vs. normal traffic).
- **K = 3** offers an **interesting compromise** in terms of separation, with its silhouette score remaining satisfactory.
- **K = 10** aligns with the **10 classes** present in the dataset, allowing for finer classification that reflects the diversity of the dataset.

The final choice of K nevertheless depends on specific objectives (general detection vs. detailed classification) and the desired granularity.

4.1.4. In-depth Analysis

In order to evaluate the relevance of the results obtained with the *K-Means* algorithm, we analyzed the distribution of connection types within each cluster for the optimal values $K = 2$, $K = 3$, and $K = 10$. The results show a strong limitation of the algorithm in effectively separating network flows into distinct categories (attacks vs. non-attacks).



(a) $K = 2$ (b) $K = 3$ (c) $K = 10$

Figure 41: Distribution of connection types in each cluster for $K = 2$, $K = 3$, and $K = 10$.

For $K = 2$ (Figure 41a), the data shows a strong concentration of benign flows (*Benign*) in *Cluster 1* (83,906 connections). However, this cluster also contains all types of attacks, although their proportions are relatively low. *Cluster 2*, on the other hand, also includes benign flows, but also several types of attacks (*Generic*, *DoS*, etc.). This distribution reflects poor separation between attack and non-attack flows.

With $K = 3$ (Figure 41b), the new *Cluster 3* contains a very low number of connections, indicating underutilization of this cluster. *Clusters 1* and *2* continue to present a significant mix of benign connections and various types of attacks. This confirms that adding a third cluster does not improve the separation of categories.

For $K = 10$ (Figure 41c), the distribution becomes more granular, but confusion persists. Several clusters (*Cluster 1*, *Cluster 4*, *Cluster 6*, etc.) contain both benign connections and various types of attacks (*DoS*, *Exploits*, *Fuzzers*, etc.). This fragmentation makes it difficult to interpret the clusters and prevents a clear separation of network flows into distinct categories.

In conclusion, the results show that the *K-Means* algorithm fails to effectively separate network flows into coherent clusters. Whether with $K = 2$, $K = 3$ or $K = 10$, the clusters contain significant mixtures of benign connections and attacks, which limits the practical usefulness of this grouping for applications such as intrusion detection.

4.2. GMM (Gaussian Mixture Model)

The *Gaussian Mixture Model* (GMM) is a probabilistic approach for *clustering*, which models each cluster as a combination of Gaussian (normalized) distributions. Unlike *K-Means*, GMM takes into account the probabilistic distribution of the points and assigns to each sample a probability of belonging to a given cluster.

4.2.1. Results and Silhouette Score.

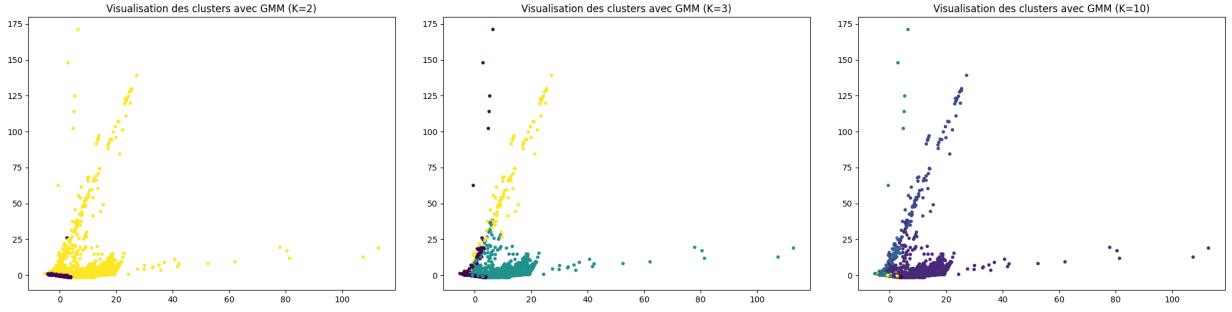
We evaluated the GMM model for $\{2, 3, 10\}$ components (clusters) and calculated the *silhouette score* for each partition. The following results were obtained:

- **K = 2** : Silhouette Score = 0.4555
- **K = 3** : Silhouette Score = 0.1183
- **K = 10** : Silhouette Score = 0.3749

As with *K-Means*, the value of $K = 2$ offers the best silhouette (0.4555), suggesting two large coherent groupings. However, for fine classification related to the 10 classes of the dataset, we also examined the configuration $K = 10$, which achieves a score of 0.3749, lower but consistent with a finer granularity.

4.2.2. Cluster Visualization.

Figure 42 presents the visualization of GMM clusters projected in two-dimensional space, applying a *PCA* (2 components). It compares the partitions obtained for $K = 2$, $K = 3$, and $K = 10$.



(a) **GMM with** $K = 2$ (b) **GMM with** $K = 3$ (c) **GMM with** $K = 10$

Figure 42: Visualization of clusters with *Gaussian Mixture Model* for different values of K (2D PCA projection).

4.2.3. Analysis and Interpretation.

- **K = 2** : With a silhouette score of 0.4555, we identify two large well-differentiated groups, one of which is significantly denser around low values (see Figure 42a).
- **K = 3** : The silhouette drops to 0.1183, indicating a strong overlap between some clusters (Figure 42b).
- **K = 10** : The score of 0.3749 suggests more fragmented clusters, consistent with a higher granularity (Figure 42c). This configuration may still be interesting to approach the logic of a classification into 10 categories, at the expense of poorer overall separation.

Overall, the *Gaussian Mixture Model* offers a probabilistic alternative to *K-Means*: it can be more flexible in situations where the shape or density of the clusters is not strictly spherical. However, the drop in the silhouette score at 3 components illustrates the difficulty of obtaining a coherent segmentation on complex datasets, especially when minority or very spread-out clusters are present.

4.3. DBSCAN

The *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*) algorithm detects clusters based on the density of points, allowing it to identify complex shapes and distinguish isolated points (*noise*) when they do not have a sufficiently dense neighborhood. In our implementation, we set `eps` $= 1$ and `min_samples` $= 50$.

4.3.1. Results and Visualization

Figure 43 illustrates the distribution of labels generated by DBSCAN, projected into a two-dimensional space via *PCA* (2 components). Points labeled -1 are considered noise (*outliers*).

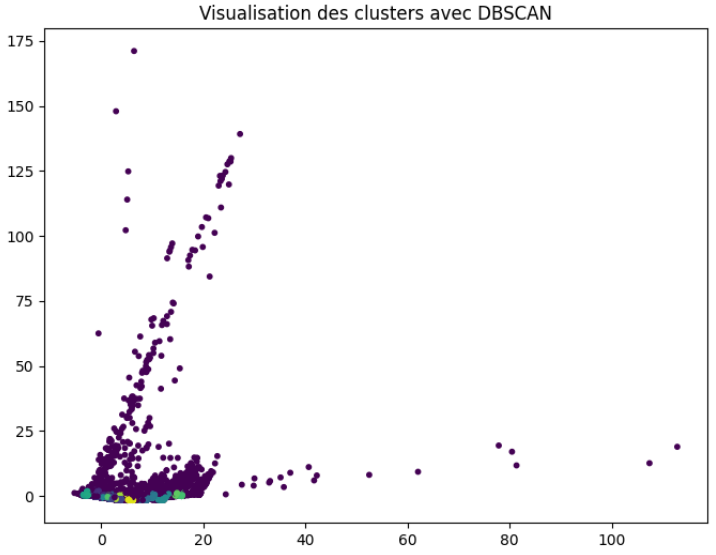


Figure 43: Visualization of clusters with DBSCAN (`eps`=1, `min_samples`=50).

The algorithm detected a **significant number of noise points**: **4366**. Furthermore, when several valid clusters are detected (and -1 is not the only label), the estimated *silhouette score* is **0.4027**, reflecting a relatively average separation between the groups.

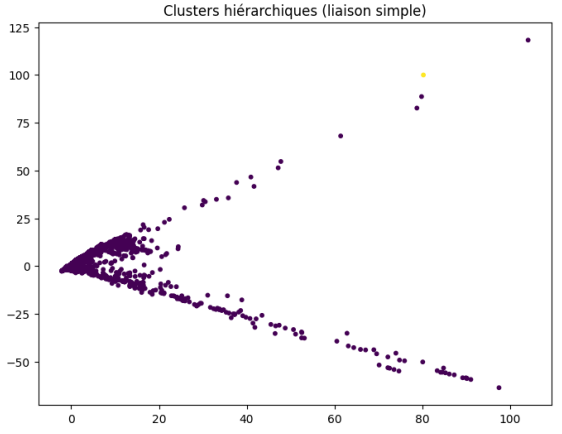
4.3.2. Analysis and Interpretation

- **A dominant cluster:** As can be observed in Figure 43, the majority of points (in purple) belong to a single extended cluster, encompassing many flows with similar characteristics (according to the first two PCA components).
- **Small groupings and noise:** However, a few small groupings (in green and blue, near the $(0, 0)$ area) can be distinguished, reflecting denser data subsets. A significant number of isolated points (-1) are located at the periphery (or sometimes elevated in the graph), signaling anomalies or atypical flows.
- **Moderate silhouette score** (≈ 0.4027): This suggests medium intra-cluster cohesion and inter-cluster separation. The distances in PCA space show that a large part of the points is aggregated in a main cluster, while others form small pockets of density or are classified as noise.
- **Parameters** `eps` **and** `min_samples`: The choice of `eps` $= 1$ and `min_samples` $= 50$ has a direct impact on the number of clusters and noise points. A higher `eps` could merge some of these small pockets with the main cluster, while a lower `eps` might further segment the dataset.

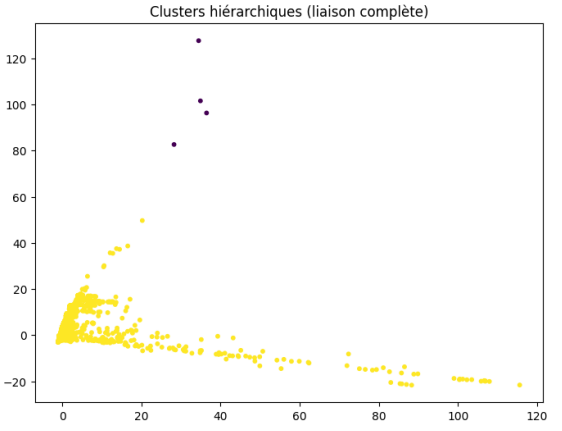
In summary, *DBSCAN* allows us to identify a major cluster that seems to correspond to the usual flows, as well as several more restricted dense groupings and a notable fraction of isolated points (4366).

4.4. Hierarchical Clustering

4.4.1. 2 Clusters



(a) **Single Linkage**

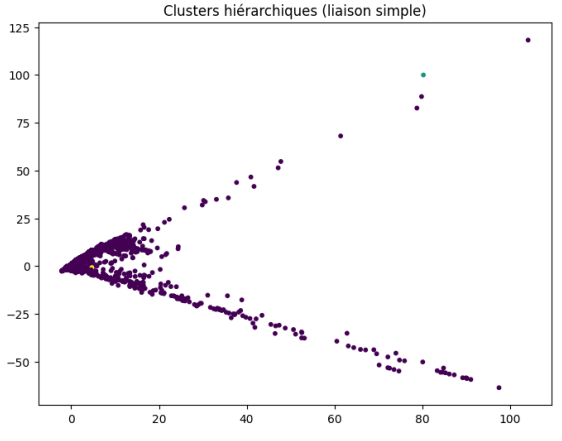


(b) **Complete Linkage**

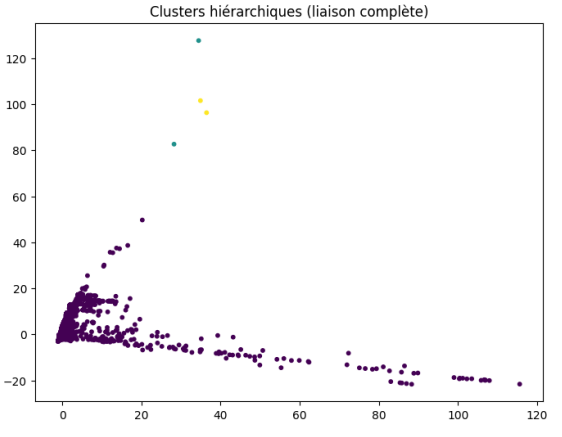
Figure 44: Hierarchical Clusters for 2 Clusters with Single and Complete Linkage.

For the configuration with 2 clusters, neither single linkage (Figure 44a) nor complete linkage (Figure 44b) effectively segments the data. The majority of the points are assigned to a single cluster, with only a few isolated points forming a second cluster. This result suggests that the distances used for merging the clusters do not adequately capture the structure of the data, making this approach uninformative in this context.

4.4.2. 3 Clusters



(a) **Single Linkage**

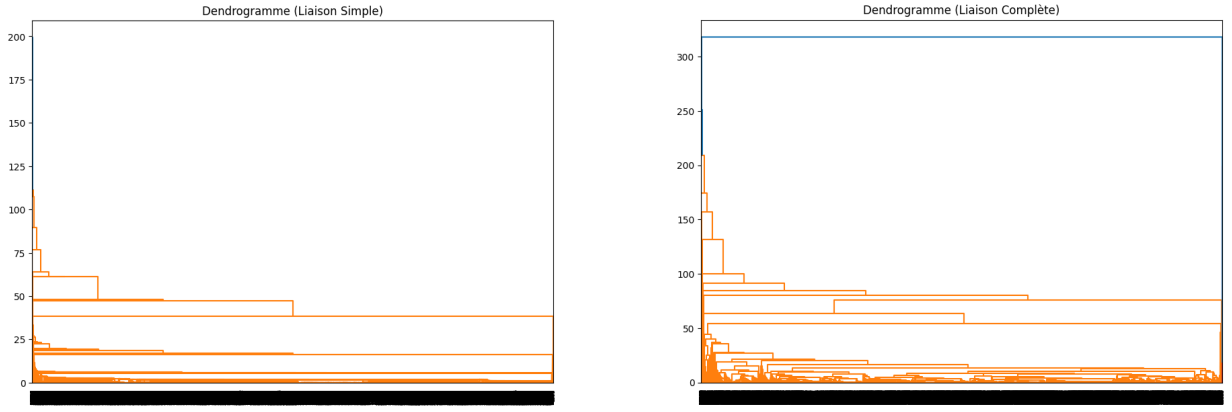


(b) **Complete Linkage**

Figure 45: Hierarchical Clusters for 3 Clusters with Single and Complete Linkage.

With 3 clusters, the observations remain similar. The single linkage method (Figure 45a) assigns almost all points to the same cluster, while complete linkage (Figure 45b) also fails to identify any significant groupings. These results indicate that the hierarchical clustering methods are not suitable for this particular dataset, due to the low differentiation among inter-point distances.

4.4.3. Dendrograms



(a) **Dendrogram with single linkage** (b) **Dendrogram with complete linkage**

Figure 46: Dendrograms of the clusters with single and complete linkage.

The dendrograms (Figures 46a and 46b) confirm this observation. The inter-cluster distances remain low throughout the aggregation process, indicating that the points are very close to each other in the data space. This explains why the single and complete linkage algorithms fail to produce meaningful groupings. The analysis of these dendrograms shows that significant breaks, necessary to determine distinct clusters, are not present in this dataset.

5. Conclusion

Overall, the work presented in this article reflects the structural and behavioral complexity of network flows and illustrates the inherent challenges in their detection and large-scale classification. Exploratory data analysis (EDA) has highlighted the most discriminative features (*Flow Duration*, *Fwd Seg Size Min*, *Bwd Packets/s*, etc.) to differentiate, on one hand, benign traffic from attacks and, on the other hand, the different categories of attacks themselves. Supervised machine learning models (XGBoost, LightGBM, CatBoost, etc.) have shown excellent performance for binary identification and multi-class classification, with a slight edge for XGBoost and ensemble methods (Stacking, Voting, Bagging). Hyperparameter optimizations via `Optuna` have enhanced precision and recall, underscoring that fine-tuning remains essential to get the most out of these algorithms.

From an unsupervised perspective, clustering approaches (K-Means, GMM, DBSCAN, and hierarchical methods) have proven less conclusive in effectively separating the classes of the `CIC-UNSW-NB15` dataset. Although K-Means and GMM provided clustering elements for two or three clusters, the distribution of flows (including attacks) within each partition remained highly mixed. Hierarchical methods, on the other hand, failed to highlight clear breaks in a highly dense and poorly differentiated space. Nevertheless, DBSCAN was able to identify density pockets and several outliers, demonstrating its ability to detect diffuse anomalies.

The complementarity between supervised and unsupervised analysis remains a promising research avenue for proactive threat detection in complex network environments. Exploring semi-supervised algorithms or active learning techniques could provide solutions for scenarios where few labeled samples are available. Furthermore, the continuous refinement of models (hyperparameter tuning, engineering new features, using techniques for balancing rare classes) is crucial to further enhance detection robustness. Ultimately, these results emphasize the importance of a multidimensional approach—combining EDA, feature selection, and model combinations—to achieve a high level of performance in the detection and classification of network attacks.