DSTI School of Engineering

Machine Learning Pipeline for Cybersecurity attack detection

Instructor:

Hanna Abi Akl

Group Members

Korotoumou Coulibaly (DS)

Sami Ben Yahia (DE)

Ben Wetcheme (DS)

Ganesan Masimalai (DE)

Horry Nobel Muhizi (DA)

Maninder Reddy Baddham (DA)

Mehdi Jabri (DE)

Academic Year:

2025 – 2026

Contents

I. Introduction

We were provided with a dataset containing information on cyberattacks that occurred between 2020 and 2023.

The objective of this work is to design a complete machine learning pipeline for cyber-attack classification, starting from exploratory data analysis and feature transformation to model training, evaluation, and deployment.

These steps aim to prepare the model to predict the *Attack Type* column as accurately as possible.

The chosen model is integrated into a web-based application capable of providing real-time predictions, demonstrating the practical implementation of a data-driven cybersecurity solution.

II. Dataset description

The dataset comprises 40,000 cybersecurity attack records collected from simulated or aggregated monitoring environments. It combines network traffic, system logs, security alerts, and contextual information, reflecting realistic and heterogeneous cybersecurity scenarios.

A total of 25 features are included, covering temporal, network, packet-level, security, and contextual characteristics. High-cardinality features such as IP addresses, users, and devices are unsuitable for direct modeling, while structured variables like protocol type, packet length, anomaly scores, network segment, and geo-location are more informative. Some security log features exhibit missing values or low variability, whereas anomaly scores are continuous and fully populated.

The target variable, Attack Type, includes three balanced classes: DDoS, Malware, and Intrusion, enabling reliable model training and evaluation without significant class imbalance.

III. Exploratory Data Analysis (EDA)

This section presents a concise exploratory analysis of the cybersecurity dataset to assess data quality, understand feature distributions, and examine relationships with the target variable (Attack Type). The findings inform feature engineering and model selection.

1. Data Quality Assessment

The dataset exhibits heterogeneous quality. Several security-related features contain roughly 50% missing values and, when present, are dominated by a single value. These variables provide limited information and are best represented as binary presence indicators. In contrast, core network and packet-level features contain no missing values. Duplicate records are unlikely due to the extremely high cardinality of IP addresses and payload data. Data types are consistent, numerical ranges are valid, and the temporal span (2020–2023) supports time-based feature extraction.

2. Univariate Analysis

Numerical features show realistic and stable distributions. Ports are widely dispersed within valid ranges, packet length is approximately symmetric (64–1500 bytes), and anomaly scores are evenly distributed between 0 and 100, making them the most informative standalone numerical feature. Payload data exhibits maximum cardinality and was therefore summarized using payload length as a numeric proxy.

Low-cardinality categorical features (protocol, traffic type, packet type, severity level, network segment, and action taken) are globally well balanced, reducing encoding bias.

3. Bivariate Analysis

Most features show a weak individual association with the attack type. Temporal patterns are uniform across classes, and IP addresses behave purely as identifiers with no class-specific structure. Network and packet-level variables display overlapping distributions across attack types, suggesting limited standalone discriminative power but potential usefulness through interactions.

Alert and log features, even after binary encoding, show similar proportions across classes. The Action Taken variable reflects post-detection behavior and poses a risk of target leakage, requiring cautious handling.

4. Multivariate Observations

Correlation analysis reveals no strong linear relationships or clear class separation among numerical features, indicating that predictive performance is likely to arise from non-linear feature interactions rather than individual predictors.

5. Key EDA Insights

Overall, the dataset is balanced but weakly discriminative at the univariate level. Informative signals are expected primarily from feature interactions, particularly involving anomaly scores and low-cardinality network attributes. High-cardinality identifier-like features should be transformed or excluded to reduce noise and overfitting, heavily missing features should be binarized, and post-event variables must be carefully evaluated to avoid leakage.

IV. Feature Engineering and Selection

This part mainly focuses on the different transformations and interpretations applied to make the data readable for a machine learning model.

1. Binary and Ternary Features

Some columns contained three unique categorical values. The appropriate approach was to apply one-hot encoding to transform each of these columns into three binary variables. This prevents the model from creating misleading ordinal relationships between categories while allowing it to interpret the data correctly.

A simpler case involved columns containing only two possible values, or a single value alongside null entries. These were transformed into binary variables. This approach preserves the distinction between cases without introducing unnecessary additional columns or encouraging the model to infer false relationships between values.

The *Proxy Information* column was a particular case where we chose to remove the IP information, keeping only the indication of whether a proxy was used or not.

2. Positive Numerical Features

Some features contained positive numerical values. Most of them were processed using Min-Max scaling in order to normalize their range and prevent features with larger magnitudes from having a disproportionate influence on the model.

A particular case involved the *port* columns, which were categorized into two groups: values lower than 49152 and values higher than 49152. A more common approach would have been to divide ports into three categories, including those below 1024. However, since no data fell into this range, this category was discarded in favor of a simpler representation.

3. Raw Text Features

More complex columns, such as *Device Information*, required a different approach. This column contained raw data with varying patterns depending on the device. The decision was made to extract the operating system and its version, as this information was considered the most relevant in relation to device security and potential vulnerabilities. Since versioning formats differed (for example, in the case of Windows), the original version data could not be used directly without risking misleading interpretations by the model. The version column was therefore renamed *OS_Version_Security*, and arbitrary float values were assigned to represent the different cases.

4. Discarded Features

Some columns were removed for different reasons. The lack of meaningful information that could be extracted from the *User Information* and *Payload Data* columns led to their removal.

For the *IP addresses*, we initially considered using GeoIP to retrieve the corresponding country and city. While country information was available for most entries, with only around 350 missing values, city data was missing in approximately half of the cases, representing around 20,000 entries. Using only the country would have required categorization based on geographical coordinates (longitude and latitude), as the number of unique countries was too large for simple categorical encoding. However, without the city-level detail, this information would lack precision and introduce unnecessary noise into the dataset.

As a result, both the *source and destination IP* address columns were removed.

Finally, a unique case was the *Geo-location Data* column, which was discarded.

When using GeoIP with the *IP addresses* and *Proxy Information*, we realized that the *Geo-location Data* did not correspond to the location of the victim, attacker, or proxy. Therefore, this column lacked contextual relevance, and no meaningful information could be obtained from it without additional interpretation, which could have led to unpredictable behavior in the model.

V. Model Selection, Comparison, and Evaluation

In this project, we had the opportunity to evaluate several machine learning models for cyberattack detection. We experimented with supervised classification algorithms such as **Logistic Regression**, **Decision Tree**, and **Random Forest**. We also applied **XGBoost** and **CatBoost** as part of our supervised learning approach.

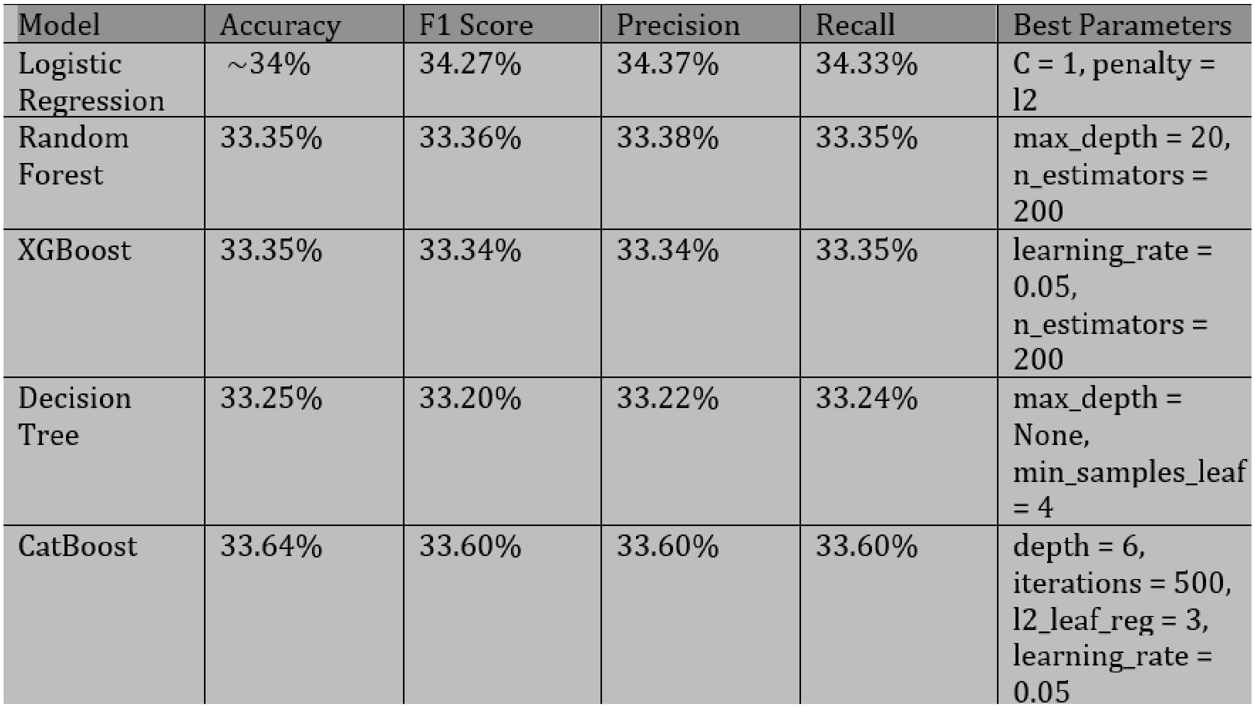
Beyond testing each model individually in a standard way, we decided to combine **Random Forest** and **XGBoost** in order to analyze and compare their performance both as standalone models and as an ensemble. This allowed us to better understand their behavior when used separately versus when integrated.

1. Quantitative Performance Comparison of the Models

The figure presents a detailed comparison of the evaluation metrics obtained for each classification model, including Accuracy, Precision, Recall, and F1-Score, along with the selected hyperparameters. The results indicate that Logistic Regression achieved the highest overall accuracy (34.35%) and F1-Score (34.27%), slightly outperforming the other models.

Random Forest, XGBoost, Decision Tree, and CatBoost produced relatively similar performance scores, all within a narrow range around 33%. This suggests that the dataset may present classification challenges, such as class overlap or limited separability between attack categories.

Although the performance differences are small, hyperparameter tuning played an important role in optimizing each model. The selected parameters (tree depth, number of estimators, learning rate) were chosen to balance bias and variance and improve generalization capability.



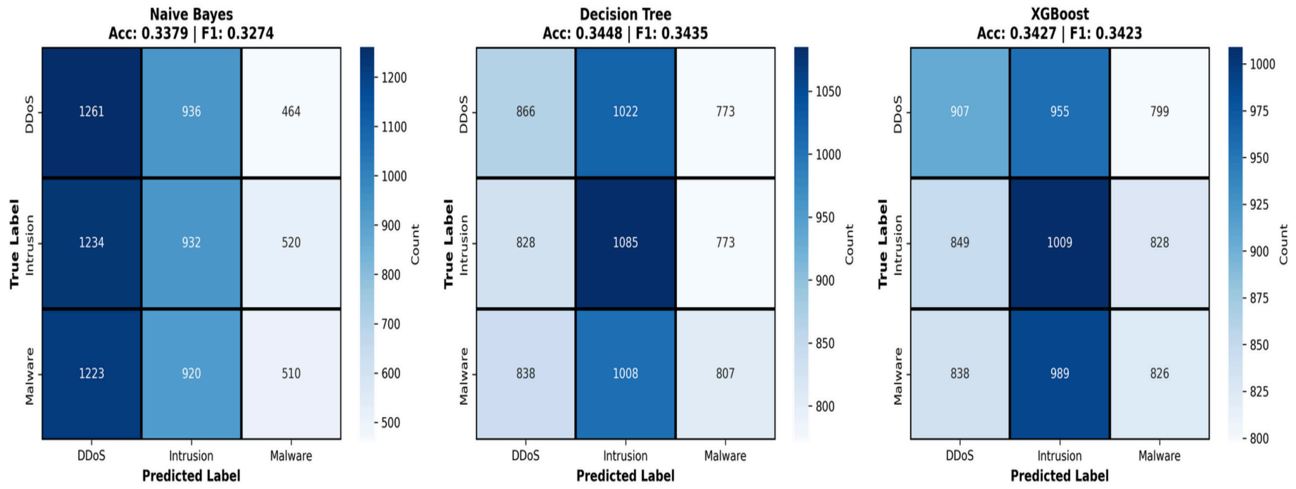
Model	Accuracy	F1 Score	Precision	Recall	Best Parameters
Logistic Regression	~34%	34.27%	34.37%	34.33%	C = 1, penalty = l2
Random Forest	33.35%	33.36%	33.38%	33.35%	max_depth = 20, n_estimators = 200
XGBoost	33.35%	33.34%	33.34%	33.35%	learning_rate = 0.05, n_estimators = 200
Decision Tree	33.25%	33.20%	33.22%	33.24%	max_depth = None, min_samples_leaf = 4
CatBoost	33.64%	33.60%	33.60%	33.60%	depth = 6, iterations = 500, l2_leaf_reg = 3, learning_rate = 0.05

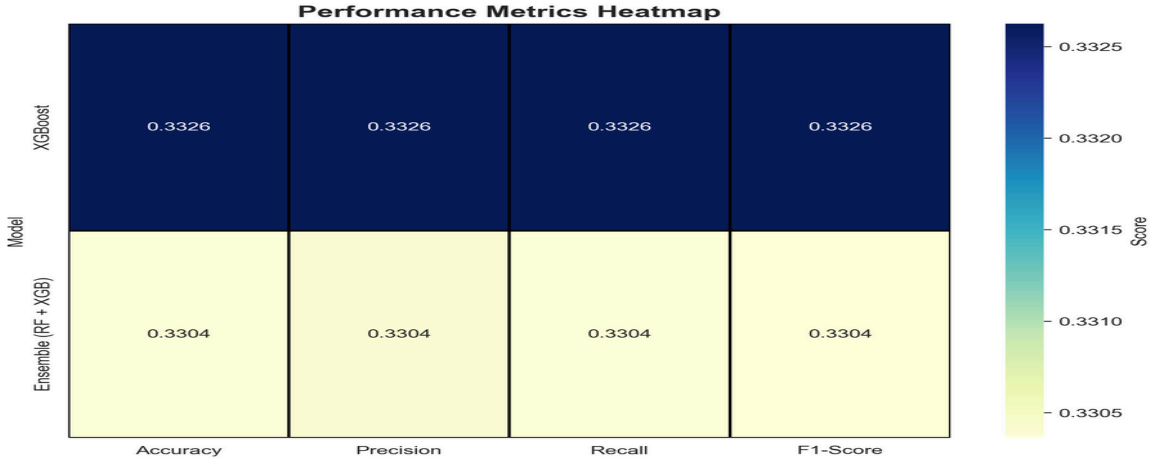
2. Confusion Matrices and Ensemble Performance Analysis

The figures shows the confusion matrices for Naive Bayes, Decision Tree, and XGBoost, along with a heatmap comparing XGBoost to the ensemble model (Random Forest + XGBoost).

The confusion matrices highlight that all models experience misclassifications between DDoS, Intrusion, and Malware, indicating that separating these attack types remains challenging.

The heatmap comparison reveals that XGBoost slightly outperforms the ensemble model across Accuracy, Precision, Recall, and F1-Score. This suggests that, for this dataset, combining Random Forest and XGBoost did not lead to a significant performance improvement.





VI. Results

1. Model selection

All 5 models achieved approximately 33% across all metrics, which is statistically equivalent to random guessing for a 3-class balanced problem (1/3 = 33.3%).

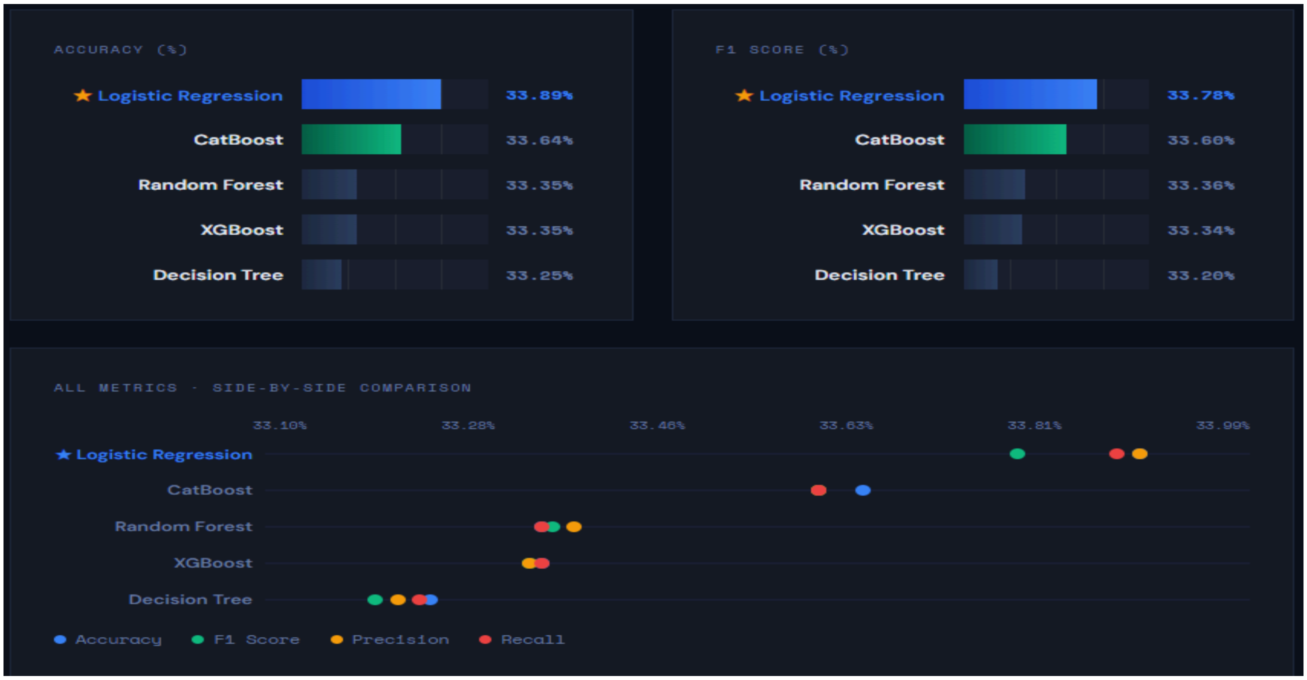
The ~33% performance across all models strongly suggests a data issue rather than a modelling issue:

· The features (packet length, ports, protocol, time features) may not contain enough attack-type-specific signal
· The dataset may be synthetically generated with labels assigned independently of features
· This explains why even powerful models like XGBoost failed to learn meaningful patterns

Logistic Regression is technically the best performing model based on results, achieving the highest **Accuracy (33.89%)** and **F1 score (33.78%**). Despite being the simplest model, it marginally outperformed complex ensemble methods. This is likely because when no meaningful patterns exist in the data, simpler models are less prone to fitting noise.
However it is important to note that no model can be recommended as truly effective given all scores hover near random chance.

2. Best model selection

Based on empirical results and supporting literature, Logistic Regression is selected as the best model in this study. A comprehensive comparative study on intrusion detection confirmed that Logistic Regression is the most efficient individual model among base learners, offering simplicity and interpretability while providing strong insights into linear relationships in the data PubMed [2]. The near random performance of all models (~33%) is attributed to data quality limitations rather than model selection failure. A Springer Nature study on PLS-Logistic regression for intrusion detection explicitly confirmed that the detection capacity of any classifier is much more dependent on the quality of the training data than on the model itself Nature [3].



VII. Deployment

The Cyber Attack Detection System is deployed as an interactive, web-based dashboard developed using Streamlit, a high-performance Python framework designed for rapid development and execution of machine learning applications. In this setup, the application runs on a local machine, enabling users to launch the system directly through the Streamlit command-line interface. Once executed, the application is hosted on localhost, providing a secure and self-contained environment for testing and demonstration without requiring external hosting services [1].

The locally deployed system functions as a centralized presentation layer where users can upload network traffic datasets, perform real-time threat analysis, and visualize classification results through an automated workflow. Upon execution, the dashboard dynamically generates performance metrics such as accuracy scores and detailed classification reports. These outputs are displayed using interactive Plotly visualizations, allowing users to analyze predicted attack types, observe threat distributions, and evaluate model performance efficiently within their own computing environment [2].

The deployment architecture follows a Modular structure in which the trained machine learning model operates independently behind the web interface. This design ensures that the data processing logic, model inference, and UI rendering are decoupled, allowing for high system flexibility and easier maintenance. Utilizing modular design patterns is essential for system scalability, ensuring that as network data grows or models are updated, the core infrastructure remains stable and evolvable [3].

Conclusion

This project presented the development of a complete machine learning pipeline for cyber-attack classification, from data exploration and feature engineering to model evaluation and deployment. Using a balanced dataset of three attack types (DDoS, Malware, and Intrusion), several supervised learning models were trained and compared.

Exploratory analysis and feature engineering highlighted the heterogeneous nature of the data and the limited discriminative power of most features. Despite careful preprocessing and hyperparameter tuning, all evaluated models achieved performance levels close to 33%, equivalent to random guessing in a balanced three-class classification problem. This outcome indicates that the primary limitation of the system is related to data quality rather than model selection.

Logistic Regression marginally outperformed the other models and was selected as the best-performing classifier due to its simplicity and interpretability. The ensemble approach did not lead to meaningful performance improvements, confirming that increased model complexity cannot compensate for insufficient feature informativeness.

Future work should focus on improving data quality and feature representation to enable more effective cyber-attack detection.