Network Traffic Classification Pipeline: Leveraging Fair and Structured Model Independent Methodologies

Ricardo Oliveira
pg54177@alunos.uminho.pt
University of Minho
Braga, Portugal

Rita Vaz
pg53642@alunos.uminho.pt
University of Minho
Braga, Portugal

Abstract

There have been many attempts to improve firewall performance through the use of machine learning but, while there are some studies with promising results, replicating and comparing different models presents a serious challenge. Despite the existence of tools that allow users to build, train and evaluate machine learning models, there's a lack of a clear and consistent pipeline among models of this specific field. In this study, we focus on developing a framework that supports both online and offline training methods for network traffic classification using machine learning. Through a foundational review of existing approaches, we identified several key challenges compromising advancements in the field: the difficulty of fair comparisons among models, the lack of consistent methods from different studies, and the limited availability of open-source code and datasets. This work bridges critical gaps in research by enabling seamless integration of offline and online anomaly detection methods, providing a versatile and reproducible tool for advancing machine learning-based cybersecurity solutions.

Keywords

Firewall, Machine Learnig, Pipeline, Network Traffic, Classification, Online, Offline, Framework, PCAP

1 Introduction

In today's world, cyber attacks are becoming more prevalent and firewalls often serve as the first line of defense to protect critical assets. The effectiveness of a firewall is highly dependent on its ability to accurately distinguish normal network traffic from malicious activity. Traditional rule-based systems, while effective against known threats, often fail to address emerging or sophisticated attacks.

Machine learning (ML) offers a promising alternative by enabling firewalls to detect patterns in network traffic that static systems might overlook. Recent advances in ML techniques have shown significant potential for improving firewall performance in areas such as anomaly detection and traffic classification. Despite this promise, the field faces persistent challenges that hinder progress.

- Lack of Reproducibility: Studies often fail to provide accessible datasets, reusable code, or clear methodologies, making it difficult to replicate or validate the results.
- Inconsistent Comparisons: The absence of standardized processes for training, evaluating, and comparing ML models complicates benchmarking and knowledge transfer.
- Imbalanced Focus: Research predominantly explores offline training methods, with limited attention to online learning and real-time applications.

However, it is important to note that this is not a review of the literature but rather a study aimed at identifying the best possible approach to address our specific problem. Our goal is to create a pipeline for network traffic classification that allows for consistent training of distinct machine learning models, using the available data, and their fair evaluation, independent of the learning technique (online or offline).

This implies proposing solutions for current problems such as the lack of process, reproducibility, and usage for the developed models. Taking this into account, the project in hands aims to make the following contributions:

- Propose a solid pipeline that allows the scientific community to train, test, compare and deploy models for network traffic classification;
- A consistent method that allows for the reproducibility of results, empowering both individuals, to test, use and learn from the models, and future researchers to further comprehend, compare to existing knowledge and/or improve upon it;
- Fair and consistent model comparison across the various techniques used when training the models. This includes taking into account not only the features and parameters used, but also the learning methodology used by the models (online and offline), understanding it's relevance.

With this taken into account, the final pipeline focuses on improving the already existent methodology, bringing a greater focus to reproducibility and inclusion of different techniques. The gap in research drawing comparisons between online and offline learning techniques for anomaly detection in network traffic, to the best of our knowledge, is still widely unexplored, leaving questions of its benefits and drawbacks, along with how to fairly evaluate either one. Unlike other studies, our methodology consolidates the steps to create a fully functional model, independent of the learning process or paradigm (supervised, unsupervised, and reinforcement), that can be easily recreated and fairly compared to results of other investigations in the field.

This paper is structured as follows: The *Related Work* section reviews key articles, summarizing their objectives and conclusions, as well as possible shortcomings. Next, we present the *Pipeline Definition*, where we outline the design of our pipeline, describing its steps along with several open-source machine learning frameworks to understand what could be beneficial. Moreover, the *Proof of Concept* section showcases the real application of our framework through its various phases, how it can be implemented, and the key points of each step. After this point, we do a *critical review* of

our proposed solution, highlighting its benefits and limitations, discussing the *challenges* faced and how they were handled. We finish this paper with the key takeaways from the project, its *relevance* when facing the challenge of network traffic classification, and the *impact* we hope it achieves moving forward.

2 Related Work

In recent years, Machine Learning (ML) techniques have been widely adopted to improve current cybersecurity practices, particularly for analyzing and classifying network traffic. This is essential for firewalls to identify malicious activity effectively. Despite promising improvements, a consistent methodology has yet to be defined for training, evaluating, and comparing models. This lack of standardization compromises reproducibility and scalability, critical requirements for advancing the field.

From a comprehensive review on the current state of the field, we identified a strong trend towards offline processing methods, with roughly 85% of the studied papers focusing exclusively on offline approaches, about 8% addressing both training methods, and only roughly 6% exclusively addressed online methods.

Although sparse, comparative studies of online machine learning (OL) and offline machine learning approaches provide a clear distinction between strengths and limitations in cybersecurity applications. For instance, in the study *A Comparative Study on Online Machine Learning Techniques for Network Traffic Streams Analysis* [35], OL's ability to process real-time data streams is highlighted as it contributes to addressing challenges such as concept drift and class imbalance, which are critical for dynamic environments like live networks.

Empirical studies provide contrasting insights into online and offline approaches. *Assessing Machine Learning Techniques for Intrusion Detection in Cyber-Physical Systems*[33] exposes that offline models excel when utilizing predefined attack signatures, whereas online models tend to respond better to dynamic scenarios with emerging threats. *Empirical Analysis of Data Streaming and Batch Learning Models for Network Intrusion Detection*[5] further supports the advantages of online learning, demonstrating that streaming algorithms like Hoeffding Tree and OzaBagAdwin outperform batch learning in scalability and real-time adaptability for detecting attacks such as denial-of-service and phishing.

Along with these efforts, Deep Learning has also been used to analyze firewall logs [37]. Using Deep Neural Networks (DNNs) and Recurrent Neural Networks, the study *Deep Learning for Unsupervised Insider Threat Detection* [37] focuses on detecting insider threats in real time based on users' network behavior. However, its reliance on anonymized synthetic datasets limits both its reproducibility and broader applicability.

Delving deeper into the importance of the Datasets being utilized in current research, *Anomaly Detection in Networks Using Machine Learning*[25] highlights crucial challenges such as obtaining relevant samples containing real traffic in the current landscape of cybersecurity, mostly due to privacy concerns and regulations. Along with *Data-Driven Network Analysis for Anomaly Traffic Detection* [9], a comprehensive review was conducted on widely used datasets (such as DARPA 98, KDD 99, and CICIDS 2017) reviewing their ability to accurately portray networks and attacks. The importance of the data used is further explored in *Comparative Analysis of Anomaly Detection Approaches in Firewall Logs*[24]. This paper emphasizes the need for real-world traffic datasets and proposes synthetic anomalies in order to simulate attacks. However, it lacks public code or datasets, limiting reproducibility. Lastly, *Semi-Supervised Learning for Anomaly Traffic Detection via Bidirectional Normalizing Flows* [18] introduces a three-stage framework that generates pseudo-anomalies using normal data, reducing reliance on labeled datasets. This innovative approach, with an additional validation on unseen data, highlights the robustness of the method, despite a slight performance drop.

Additional challenges in the fields of network security are addressed in *Anomaly Detection of Policies in Distributed Firewalls*[10] and *The Analysis of Firewall Policy Through Machine Learning*[38]. By researching conventional firewall and their policies, the former studies analyze limitations of static rules in detecting complex anomalies. While our work focus on the traffic itself, the mentioned papers offer strong insights into feature selection and preprocessing, discussing issues such as the impact of the training dataset size on performance.

Collectively, these studies highlight the diverse approaches and challenges within ML-based network traffic classification. Insights from these studies contribute to the design of the pipeline, which will integrate preprocessing, model training, and real-time classification to ensure robust process both for comparing and evaluating the performance of various models across multiple metrics. In addition, these studies underscore the need for a structured approach, employing dataset selection, the strengths of both online and offline methods, and a comprehensive evaluation, in order to achieve effective reliable models that can adapt to and evaluate real network traffic.

3 Current Limitations

Throughout the study of the different approaches and solutions to the problem in hands, we noticed several key factors that needed to be improved. In this chapter, we will go through the three main aspects that, if improved, could lead to a better and more consistent trend in research. These key aspects are:

- Pipelines & Frameworks: this includes exploring the methodology used, its process and what limitations does it pose to future project;
- Datasets: giving the volatile nature of network traffic and cyber-attacks, having representative datasets is key to an accurate model training;
- Code Bases: while often this resource is missing or not public, code bases are fundamental to research as they help support the results obtained and guide future improvements in the area.

Finding the limitations for these components across various recent and relevant studies, is extremely important to outline our solution when building the pipeline and proof of concept. Since one of the main goals of the project is its relevance for further improvement of methodology and reproducibility, independent of the model or dataset, we will focus mainly on these considerations when analyzing the limitations found.

3.1 Pipelines & Frameworks

Throughout our research, we analyzed various frameworks developed for anomaly detection in network traffic. While many of them follow similar steps, they often differ in their level of abstraction, preprocessing techniques, and suitability for online or offline scenarios. These differences expose key limitations such as:

- Lack of Generalization: Most of the studied frameworks are developed to answer a specific problem, rendering them too specific;
- Overly Abstract: Conversely, efforts to generalize the process tend to lack practical detail;
- Limited Model Support: Most frameworks focus exclusively on either online or offline training.

Frameworks like *Anomaly Detection in Networks Using Machine Learning* [25] and *Anomaly Detection of Policies in Distributed Firewalls Using Data Log Analysis* [10] present specific, non-generalizable pipelines with a focus on offline methods, limiting their use in dynamic scenarios, hindering their broad applicability. Similarly, frameworks such as *Classification of Firewall Log Files with Multiclass Support Vector Machine* [20] and *Deep Learning for Unsupervised Insider Threat Detection* [37] are highly specific in their design, either by focusing on particular datasets or online data streams.

Furthermore, some studies such as *Improving Internet Firewall Using Machine Learning Techniques* [31] and *Machine Learning to Detect Anomalies in Web Log Analysis* [15], specify models and preprocessing steps without adequate justification, which limits both their generalization and scientific relevance. However, the lack of online model support in many of these frameworks, such as in *The Analysis of Firewall Policy Through Machine Learning and Data Mining* [38], is the predominant factor in limiting their adaptability to real-time systems. Factors like concept drift and emergence of new threats pose serious risks to models unable to quickly react, possible leading to serious compromises in the security of the network.

All these factors can be summarized, for the studied pipelines, in the following table:

Pipeline	Generalization	Offline	Online
[10]	Specific	X	
[12]	Too Abstract	?	?
[15]	Specific	X	
[20]	Specific	X	
[25]	Specific	X	
[31]	Specific	X	
[37]	Specific		X
[38]	Specific	X	
Goal	Generalized	X	X

Table 1: Limitations of existing anomaly detection pipelines.

To address this limitations, we highlighted crucial points to solve the previously mentioned problems. A well structured pipeline should have:

- Support for both online and offline anomaly detection, ensuring applicability to dynamic and static datasets;
- Modular design that abstracts preprocessing, feature selection, and evaluation metrics, making the framework adaptable to various use cases;
- Clear justifications for methodological choices to improve transparency and scientific rigor.

Such a framework would aim to standardize anomaly detection processes, foster reproducibility, and encourage collaboration within the research community. By decoupling data preparation and model training, it would facilitate the integration of new techniques, datasets, and metrics, allowing for meaningful comparisons across studies. This approach is essential for addressing the evolving nature of cybersecurity threats and network traffic while ensuring fair comparisons across different studies.

3.2 Datasets

The paper *Data-Driven Network Analysis for Anomaly Traffic Detection* [9] highlights the limitations in many of the public datasets and offers more realistic network data, accounting for diverse attack scenarios and a variety of benign background traffic. The key features emphasized in the study include:

- Size of the dataset;
- Lack of diverse traffic;
- Lack of diverse attacks;
- Unrealistic scenarios;
- Synthetic development process;
- Produce traffic by simulation;
- Network topology.

Several widely referenced datasets, along with their limitations as noted in the paper, are commonly used for network intrusion detection:

- DARPA (1998-1999) [4]: This was the first intrusion detection dataset but is now outdated. It lacks realistic network traffic and contains synthetic, simulated traffic, which makes it too predictable and not representative of real-world scenarios.
- KDD99/KDDCup99 (1998-1999) [2]: Based on the DARPA dataset, it is widely used for evaluating detection methods. However, it suffers from duplicated samples, an imbalance between attack types, and outdated attack scenarios. It also does not provide diverse benign or modern attack traffic.
- NSL-KDD (2009) [3]: This dataset aimed to remove duplicates from KDD99 but still contains imbalanced data samples and lacks newer attack types, making it less suitable for current network security evaluations.
- CICIDS2017 [1]: Developed by the Canadian Institute for Cybersecurity, it includes modern attacks such as SQL injections and brute force. However, it suffers from unrealistic simulated attacks, a highly imbalanced dataset, and duplicate samples, leading to low accuracy and high false positives in machine learning models.
- *UNSW-NB15* [29]: Created by the Australian Cyber Security Centre, offers a more comprehensive range of attack types compared to KDD99 but still lacks modern attack scenarios. In addition, the dataset was created synthetically, which limits its real-world applicability.

These limitations highlight the need for more realistic, varied, and balanced datasets that better represent current network traffic and attacks.

Other challenges we encountered in our research were the lack of publicly available datasets. This is an important issue to highlight: without accessible datasets, it becomes difficult to compare results or test code from models that may be available if the specific datasets used in prior studies are not. Furthermore, the datasets currently available to us are often incompatible with the model implementations.

To address this, we compiled a list of the datasets most frequently mentioned in the literature reviewed.

Name of Dataset	Citation	Publicly Available
KDD CUP 99	[12], [30], [35]	Yes
HttpParams	[12], [34]	Yes
UNSW-NB15	[5], [35]	Yes
Internet Firewall	[8], [20], [26], [31]	Yes
CSIC 2010	[13], [14], [34], [39]	Yes
CICIDS 2017	[25], [32]	Yes
DRUPAL	[13], [14]	No

Table 2: Datasets, their Citations, and Availability

In recent years, some organizations have taken steps to address these challenges by restricting the publication of papers to those that make their source code and datasets publicly available. This approach fosters community collaboration, enabling more researchers to access and build upon existing work, thereby reducing the problem of inaccessible resources. Although privacy remains a crucial concern, effective anonymization techniques can mitigate these risks since it is not necessary to know the exact origins of the data, as long as the essential characteristics of real-world traffic and attack patterns are preserved.

For future research and development, it is essential that datasets meet certain requirements to ensure their usability and relevance, including:

- Accurate representation of attacks that reflect realistic and up-to-date scenarios;
- Diversity in both benign traffic and attack types, covering a wide range of protocols and threat landscapes;
- Adequate balancing to avoid skewing results due to class imbalances or duplicated samples;
- Public availability, assuring reproducibility and independent evaluation.

All these recommendations will contribute both to the success of the project, as well as further combating a usual deficit in the field.

3.3 Code Bases

From the papers studied, only approximately 17% had publicly available source code, already limiting the ability to verify existing results and make fair comparisons of the research.

Building on these issues, many existing projects are often incomplete or unsuitable for real-world applications. An even greater challenge lies in the codebase itself: not only are these repositories unfit for extension to other models or use cases, but they also contain hardcoded variables, values, and functions that render the code difficult, if not impossible, to adapt.

That said, we studied the available resources, trying to understand the processes and strategies being applied. Following trends in typical machine learning projects, all the available repositories implemented their solutions utilizing Python and its libraries, both for model training and data handling. Along with the modules and scripts, some of the projects supplied Jupyter Notebooks that, in a more user-friendly way, are responsible, mostly, for the process of data visualization and analysis of results.

Although the source code of these projects, in most cases, could not be tested, the projects of the papers *Evaluating ML-based anomaly detection across datasets of varied integrity: A case study* [32] and *Anomaly Detection in Networks Using Machine Learning* [25] provided valuable insight into the practical process.

The first one, focusing on data quality, consisted of various playbooks tasked with analyzing various datasets (also mentioned in the previous subchapter). Although the data and results visualization were improved to fit the needs of our proposed pipeline, it proved to be a good starting point for comparing different datasets and models. The main drawback, as displayed by the structure itself, is that the analysis and models are tightly linked to the data sources.

As for the latter one, the project contained two versions, python and jupyter notebook, of all the steps in the implemented pipeline. This implies that, similar to common ML workflows, it contained the steps for the preprocessing of data, model training and evaluation, as well as some data visualization. This was a major improvement from the available resources, as it defined each step clearly, an important point in this study. However, even if the implementation follows a structured pipeline, it lacks clarity and adaptability, suffering from previously mentioned problems such as hard coded labels & values (dataset specific), making it challenging to test and compare with different models/types of data.

All the research on the available code, more than providing a good practical foundation, was fundamental to the definition of some requirements for our pipeline. The objective is to learn from the experience obtained through the research while improving the process and assist future researchers in their projects. With this in mind, the proposed pipeline should:

- Have a clear structure, allowing for a clear understanding of the various phases;
- Be independent of the input data or model being developed. Instead, it should be easily extensible, defining "interfaces" to assure a seamless integration of new models or types of data;
- Have a reproducible process, allowing results to be validated and further tested/improved by the scientific community.

Assuring these three points, along with previous objectives (such as the fair evaluation of models), will contribute to both a well structured approach, a reliable pipeline, and a solid base for further improvements.

4 Pipeline Definition

The reviewed papers showcased various methodologies for developing machine learning algorithms aimed at detecting anomalies in firewall logs. However, these studies were very task-specific, focusing closely on the detection process rather than taking a comprehensive approach that can be replicated and/or improved [31] [15] [38]. This, while it might be efficient, can pose a significant challenge for further advances in the broader research field.

Moreover, studies often lacked a comparative and fair evaluation of developed models with preexisting research [12]. This includes the comparison of results from models trained with similar/different datasets, using online/offline learning techniques, different selection of features or even changes in parameters definition. This is an essential aspect that we aim to address in this project.

The first step to develop our pipeline was to create an abstract pipeline, as shown in Figure 1, which consists of five essential steps:

(1) Data Gathering: This step involves studying and selecting the appropriate datasets for model training;
(2) Pre Processing: This include the data cleaning, feature extraction and engineering, among others;
(3) Model Training: The training of the machine learning model, independent of the techniques used;
(4) Model Evaluation and Comparison: Fair evaluation and comparison of trained models;
(5) Model Deployment: Application of the trained model in real scenarios.

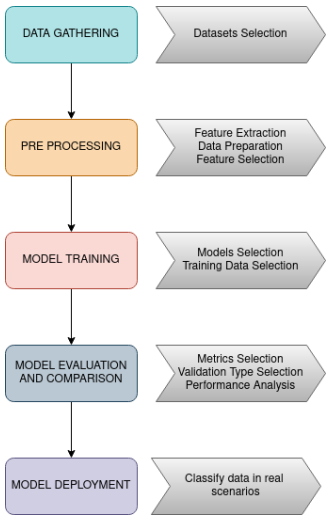

Figure 1: Base Pipeline

The gathering of data is a pivotal step in the framework, as the quality, diversity, and relevance of the collected data fundamentally shape the training process and directly influence the model's future performance. The choice of data source — whether they are network captures (pcap files), live traffic, or a combination — must align with the goals of the framework and reflect the real world context of the models being tested. Some datasets may offer rich, realistic traffic patterns that enhance model generalization, while others may lack diversity or contain outdated attack patterns, leading to limited applicability. By supporting both online and offline data collection methods, the framework accommodates for a wider range of use cases, ensuring its flexibility. This step underscores that the success of the machine learning models begins with the data itself, as it determines the relevance and reliability of the insights the models will ultimately provide.

Preprocessing is the first step for ensuring fairness in model comparison by standardizing the input data, independent of its source, making it suitable for diverse machine learning algorithms. This includes extracting and parsing features from the existing network traffic, scaling/standardizing the values, normalizing distributions, encoding categorical variables, and addressing imbalances in the dataset. These steps have the objective of removing inconsistencies that could otherwise skew results, ensuring that models are trained and evaluated under comparable conditions. Preprocessing the data aims to establish a consistent foundation, ensuring that evaluations focus on the performance of the models rather than being influenced by variability in the input data. It's important to note that, while this process takes on the task of making the training input uniform across the field, during the process of creating models, specific features may be selected and/or engineered freely in order to meet its goals.

The next logical step is the training of the various models, providing users with the flexibility to define their preferred features, hyper-parameters, and even learning processes (i.e. online/offline, supervised or otherwise). This more generalized approach ensures that the framework is adaptable to the various methodologies, empowering further creation with different machine learning techniques while adhering to a standardized process for defining the key components of model training. The main challenge for this step is the definition of the requirements all models must adhere to, the goal is to provide a solid structure for future usage in varied environments along with methods allowing for fair comparison across models.

After the models are trained, the evaluation step involves assessing and comparing them based on various metrics that can be applicable to both online and offline scenarios. This step is crucial as it ensures that the framework can accurately evaluate and compare models regardless of how they were trained, focusing on their ability to perform under realistic conditions and yielding fair, actionable insights into their strengths and weaknesses.

The final step, once the models have been trained and thoroughly evaluated, is their deployment in real-world scenarios. This implies that incoming traffic will need to undergo all the previous procedures of data preprocessing and feature selection/engineering by the model. In addition to this process, the model will then need to classify the data. In the case of online learning models, they will still be adapting to the new traffic and learning new insight from it. This means that, having a solid and well defined pipeline will play an important role on their evolution, as online models might need to regularly be evaluated and, in cases where their performance in detecting anomalous network traffic worsens, roll back to a precious state of the model. On the other hand, offline models, while staying the same during this stage, might need to undergo retraining in order to adapt to new and emerging threats, previously unaccounted for. All of this calls for a structured process that can easily be replicated and improved upon.

4.1 Tools & Frameworks

In our research, we explored several open-source machine learning frameworks with the goal of identifying helpful tools that could be integrated into a consolidated pipeline for anomaly detection in network traffic. While many of these frameworks were not specifically tailored to our use-case, they often offered features that could be leveraged to build a versatile and robust pipeline. The primary goal was to determine which of these tools could be adopted for this project while maintaining flexibility and integrating seamlessly with the complete pipeline.

Bellow we'll go through some of the tools and frameworks we came across during our investigation, along with a brief explanation of their benefits and drawbacks.

Kubeflow Pipelines and Apache Airflow provide powerful orchestration capabilities, particularly for managing complex, containerized workflows across cloud and on-premises environments. However, the Kubernetes-centric design of Kubeflow Pipelines introduced an additional layer of complexity that we deemed unnecessary for this project's objectives, as we aimed to create a straightforward pipeline accessible to users with foundational machine learning knowledge.

Airflow, in contrast, provides dynamic Directed Acyclic Graph (DAG) generation, enabling workflows to adapt to evolving data conditions. Its flexibility, combined with native Python support, advanced scheduling features, and compatibility with online training workflows, makes it well-suited for applications involving continuous data streams. However, it requires pipelines to be explicitly defined, which limits its accessibility as a universal solution for all users.

Metaflow also stood out, offering a simple Python-based API for creating machine learning flows on various platforms, including AWS, Azure, Google Cloud, and local environments. Metaflow supports "Event Triggering," enabling workflows to launch automatically upon detecting new data—an essential feature for online training scenarios. While Metaflow doesn't come with pre-defined workflows, it provides flexibility for customizing pipelines and adapting to different data sources, making it valuable for more complex, multi-environment projects. There are existing projects, such as [this template], but they lack the capability to compare online and offline models.

MLflow emerged as the strongest candidate for model visualization and comparison. Its Python-based implementation simplifies integration into existing workflows, while its support for both offline and online workflows makes it a versatile tool for diverse use cases. As an open-source platform, MLflow offers adaptability and a comprehensive environment for tracking model performance, enabling future researchers to monitor advancements in new models and compare them directly against preexisting or stable ones.

Meanwhile, frameworks like H2O 3, Weights & Biases (W&B), and Neptune.ai offer unique features for model tracking, visualization, and artifact management, particularly in scenarios where real-time prediction and monitoring are critical. Apache NiFi and nPrint were assessed primarily for data ingestion and transformation but showed limited functionality for model training and tracking, making them more suitable for preprocessing steps rather than end-to-end machine learning pipelines.

4.2 Proposed Pipeline

To develop a solid and versatile pipeline, we prioritized tools that supported adaptability across various data sources, environments, and technologies, as well as future extendability. This led to Python and its frameworks — widely used in both data science and cybersecurity — being one of the main choices. Being commonly utilized in the field of Machine Learning and Data Science, libraries like Pandas and SKLearn (among others), presented benefits not only in their capabilities, but also in their already consolidated presence in the field, further contributing to a more consitent process amongst projects.

Given these criteria, the following tools were selected for our pipeline:

- Pyshark - To capture and process network packets for real-time or offline analysis. Useful for extracting data from PCAP files and conducting network traffic analysis.
- Pandas - For data manipulation and analysis, especially for handling tabular data structures such as DataFrames.
- NumPy - For numerical computing, offering efficient array operations and mathematical functions.
- Matplotlib - For creating static, interactive, and animated visualizations in Python. Its extensive customization options make it a go-to library for plotting graphs and charts.
- SKLearn - For implementing machine learning algorithms and utilities, such as preprocessing data, splitting datasets, and evaluating models.
- Scikit-learn - Similar to SKLearn, for performing machine learning tasks, providing efficient implementations of algorithms and tools for predictive modeling and data mining.
- River - For incremental machine learning on data streams. It supports continuous learning models that can process one sample at a time.
- MLflow - For its ability to track and snapshot models independently of their learning method, as well as its seamless integration with Python and its libraries. This adaptability and strong suite of features for model performance tracking and management made MLflow an optimal choice for our project, allowing for effective model comparison and reproducibility across different machine learning techniques.

Taking everything discussed so far, we then developed our final version of the pipeline, as illustrated in Figure 2, which provides a more detailed overview of the previously discussed steps, along with a depiction of how the different technologies integrate into the different steps that compose the proposed pipeline.

With the newly created pipeline we aim to assist users in designing their own models while adhering to its structured methodology, improving not only the training workflow, but also the validation, comparison and deployment processes. This proves to be even more relevant when taking into account the scientific importance in the ability to replicate the results of presented studies.

This pipeline highlights a focal, yet less researched, topic on the field of network traffic classification/anomaly detection, the ability and importance to fairly compare between online and offline models and their effectiveness in different environments or scenarios, understanding its benefits and drawbacks. Furthermore, it presents the ability to both train and learn using previously stored network

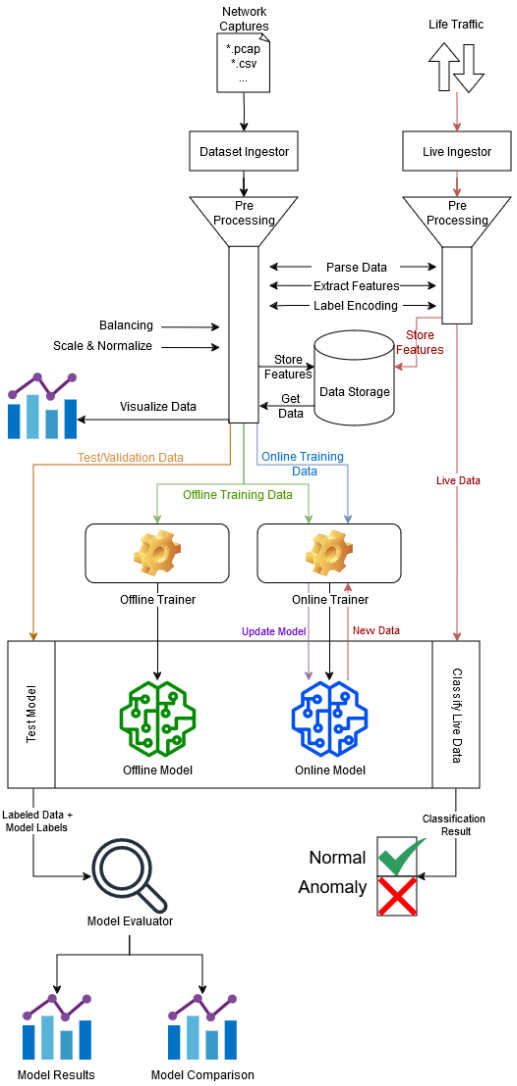

Figure 2: Proposed Pipeline

traffic, but also, the classification of live data streams (as a firewall would do) while, for online models, still improving upon the new knowledge.

5 Proof of Concept

In this chapter, we will go through the various steps of the proposed pipeline, detailing how they could be implemented and discussing the decisions made during the creation of the pipeline. This implementation takes into account the findings obtained during the review of the available research, aiming to surpass existing limitations pointed in the previous chapters.

For the implementation, and aiming to demonstrate the pipeline's value and usage, this chapter serves as a proof of concept. While the process is, and should be, more generalized, we selected relevant datasets and models in order to present a real use-case for the proposed methodology.

The full pipeline developed for this project, with the aim of further consolidating our goals, is available at [the project repository]. The different components that make up the full pipeline follow a well structured approach, promoting the modularity of the project while aligning it with the various steps of the process. Before delving deeper into the process of implementation, we present a comprehensive overview of the available code-base bellow :

- The [*data*] folder is designated for users to store input files such as PCAPs and CSVs. This will be utilized during the pre-processing stage.
- The [*docs*] folder contains comprehensive documentation to further help guide researchers and users in understanding and utilizing the proposed pipeline.
- Within the [*notebooks*] directory, two Jupyter notebooks are provided: one outlines the full pipeline process, while the other is designed for comprehensively visualize the data, independent of the current stage of development.
- The [*src*] folder is the main focus for future improvement. It is organized into multiple subdirectories, corresponding to the framework's steps, and contains modular components for [*preprocessors*] and their [*databases*] (when applicable), [*visualization modules*], [*models*], and [*evaluators*]. Each subdirectory contains both the template that defines the necessary functions for each component, allowing users to follow a structured approach that can easily be integrated with the already existing pipeline.
- Finally, the [*tests*] directory includes several Python scripts that perform various tests to verify the framework's functionality and performance.

In the following sections, we will go through the pipeline, step-by-step, further consolidating its structure and the decisions made. For each component, a comprehensive understanding of its implementation is provided in a effort to foster the reproducibility of this pipeline, however, specific details such as the technical implementation of the different modules and parameters utilized for the presented tests can be found in the [pipeline's public repository].

5.1 Data Gathering

The first step of the pipeline is the selection of a representative dataset for the case we are trying to study. Based on the research conducted, for our implementation, we selected a modified version of the UNSW-NB15 dataset [29], a widely recognized dataset designed for evaluating Network Intrusion Detection Systems (NIDS). Originally developed by the Australian Centre for Cyber Security (ACCS) at the University of New South Wales in 2015, this dataset addresses several of the limitations of existing datasets in the field of cybersecurity, such as the inclusion of real traffic and low footprint attacks.

The UNSW-NB15 dataset is publicly available, allowing researchers and practitioners to access and utilize it for their studies and developments in network security. The original dataset contains a diverse range of network traffic data, including both normal and malicious activities. It was generated using a combination of real-world traffic and synthetic attacks, providing a comprehensive representation of contemporary network environments. The dataset includes over 2.5 million records, featuring various attack types

such as DoS, probing, and exploitation attacks, as well as normal traffic patterns.

However, in order to better align with our framework's objectives, we used a modified and labeled version of UNSW-NB15 that separates the traffic captures (.pcap files) into distinct files containing either only normal data or specific types of attacks. For instance, some captures include solely normal network traffic, while others are dedicated to specific attack types, such as Denial of Service (DoS). This clear separation was a key factor in selecting this dataset, as it allows for focused training and evaluation scenarios. It simplifies the process of training models within environments containing only normal traffic and facilitates testing against targeted attack scenarios, making it easier to analyze how different models respond to new and varied situations.

This separation, while not mandatory, proves to be excellent for the evaluation of trained models. Not only does it allow to train within specific environments, for example, without any irregular traffic, but more importantly, it provides a varied range of scenarios where models can be tested, that is, verify how different models respond to different and new situations.

On the other hand, this dataset may present limitations when it comes to keeping up-to-date in the current landscape of cybersecurity and how accurately it may portray real world networks. Despite this, and taking into account the goal of this paper, to demonstrate the importance and benefits of a well structured pipeline for network traffic classification, specially when comparing and evaluating models fairly, we strongly believe that its benefits vastly outweigh the existing drawbacks.

5.2 Data Preprocessing

Once the data source is defined, the next phase is to apply the preprocessing tasks in order to prepare the available data for model training and testing. One of the most important aspects to take into account is the data format, in our case, Packet Capture (PCAP) files. These, as the name suggests, represent captures of network traffic which, given the context of the project, are the most appropriate format as they allow for the same preprocessing process to be applied both in stored and live data.

The priority of this step is to convert the raw packets into data structures that can be used by the models. For this task, we utilized Pyshark in order to extract all the available fields from the network traffic into dictionaries. This includes extracting the different layers that compose a packet, as well as their inner values (for example, the source and destination ip) and converting them to the proper data types. All of these new features need to be present, independent of their presence in any given packet. This leads to the data being flattened into a table, representing absent features as missing values, so that the models can more easily use the input data, as well as select only relevant features without excluding any data points that might not have them.

To enhance pipeline performance and minimize redundant processing, the preprocessor is designed to interact with a database manager. This integration enables efficient storage and retrieval of already preprocessed data, streamlining workflows and reducing unnecessary duplication of effort. For this task, and due to the volatile structure of the packets, we chose to implement a MongoDB database, as it is well-suited for handling diverse types of data. This flexibility allows the traffic to be stored in a dictionary like structure, containing solely the existing features, and only during accesses it needs to be transformed into a table to be used by the models. This is further beneficial since models might select only a subset of features, reducing the need for runtime resources, especially during offline learning, as the whole dataset doesn't need to be retrieved.

Another important phase is the classification of the available data, preparing it for the next steps. Since the datasets were already split according by its type, we simply labeled the data into *normal* and *anomaly* according to their source. However, some datasets may not present this information and for that cases we created the label *unknown*. Even though unlabeled data may not present much relevance to supervised learning techniques, it can still be used when training unsupervised models.

All that is left for this module is the actual access to data. One of the main aspects to prioritize is the fairness of the process, assuring all the models have access to the same training and testing values. For this purpose, a pseudo-randomized function was developed to allow consistent access to the data based on a seed. This allows not only for models to obtain the exact same training data, but also to improve performance by making it so that the data only needs to be accessed once for each step of the pipeline, independent of the number of existing models. Having the dataset split into three, i.e., offline/online training and evaluation, this process also makes sure that, when "randomly" polling from the database, no data will overlap in any of the three subsets.

It is important to mention that the current module acts both as a proof of concept and guideline to the usage of the proposed pipeline. Custom preprocessors can be implemented, allowing further compatibility for different file types such as CSV, which is commonly used in machine learning for datasets due to its structure. Furthermore, the process itself of feature extraction, data cleaning, etc. can be improved or changed to meet the needs and requirements of future projects. This is possible since the preprocessor is responsible for providing the exact same data points for the different models (both during training and testing), thus ensuring the fairness of the process.

5.3 Data Visualization

The next step focuses on visualizing the collected data. Many papers in the field provide minimal detail about their datasets, often mentioning the database used, but offering little to no further analysis. However, effective data visualization is crucial at every stage of the framework, from initial exploration to preprocessing and model training. Visualization enhances our understanding of the dataset, enabling better-informed decisions throughout the process. Unfortunately, many studies overlook this step, missing the opportunity to extract valuable insights from their data.

It is essential to begin with a global analysis of the dataset, examining its size and distribution. In real-world scenarios, anomalies generally make up less than 1% of the data, as they are rare events. Therefore, subsequent steps may involve reducing the dataset to better address this imbalance. Visualization tools such as bar and pie charts can be used to depict the distribution of normal and anomalous data, helping to understand the dataset's structure.

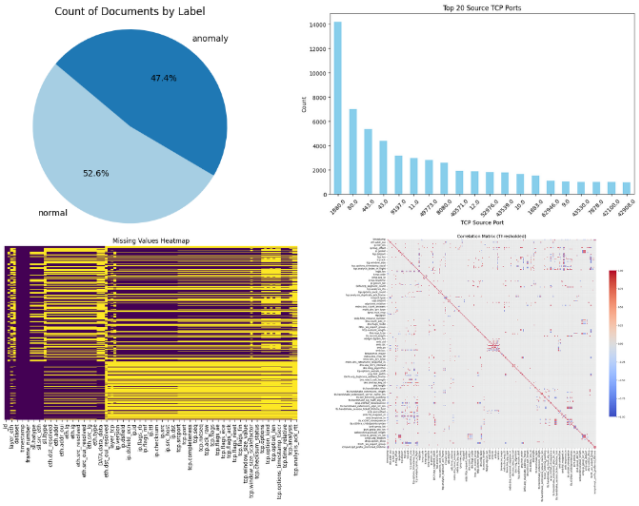

Figure 3: Data Visualization

During column-level analysis, if columns with incorrect formats are identified, they should be corrected during preprocessing. This may include converting data types or addressing formatting inconsistencies. Additionally, it is useful to analyze the common features across various datasets in order to identify similarities and differences between them. Also, identifying columns with missing values or only a single unique value can help eliminate irrelevant features that do not contribute to model training. To assist with the analyses, a function was implemented to identify columns with more than a specified percentage of missing values, creating a bitmap of the results, as well as functions to filter columns with fewer values than a defined threshold.

Drawing on insights from reviewed papers, key features crucial for network traffic analysis and anomaly detection can be prioritized. These typically include Source IP (originating traffic address), Destination IP (target traffic address), Source Port (originating port number), Destination Port (targeted port number), and Protocol (such as TCP or UDP). These features should be analyzed in detail and visualized using bar charts or similar methods to understand their importance in identifying malicious activities.

A correlation matrix is also a useful tool for evaluating the relationships between features. By analyzing this matrix, redundant or highly dependent features can be identified, aiding in the selection of the most relevant attributes for model training.

These visualization techniques are just a few examples of what can be done in this phase. The results can provide deeper insights into the dataset, influencing model performance. Thus, it is crucial to take full advantage of data visualization to extract as much useful information as possible.

5.4 Model Training

After visualizing the data, it is time to train the machine learning models for classifying the network traffic. In this proof of concept the main objective is to evaluate the process and not the models themselves. For this reason, we resorted to selecting relevant and commonly used models from the studied literature. Bellow we present the selected models, as well as their diverging characteristics, further providing insights on how the pipeline is designed to comprehensively handle the training and evaluation of different techniques.

From the analyzed papers, the most commonly used models were Support Vector Machines (SVM) [23], K-Nearest Neighbor (KNN) [17], Naive Bayes [41], Random Forest (RF) [27], Decision Tree (DT) [16], and Logistic Regression (LogReg) [19]. Although some implementations of these models were available, they often contained errors, prompting us to develop our own versions. Based on this fact, the following models were selected:

Model	Type	Mentions
Random Forest	Supervised	[6], [9], [11], [12], [21], [24], [25], [26], [31], [32], [33], [36]
Decision Tree	Supervised	[7], [8], [10], [15], [24], [28], [31], [32], [34], [35], [36]
Support Vector Machines	Supervised	[7], [10], [11], [15], [21], [24], [26], [28], [34], [36], [37],
Bagging	Ensemble	[35]

Table 3: Models in Machine Learning for Anomaly Detection

As for the features used by each model, a similar methodology was applied. During the research process we gathered the most commonly selected features for model training, along with what they represent, for an accurate representation of the current process of feature selection. The selected features are represented in Table 7.

While this features were selected for the training of all developed models, different models have the ability to select different features (within the same sample of data) according to their needs and objectives. This is further explored in the evaluation of the models, as the choice of different subsets of features might impact their performance both positively or negatively. Furthermore, each model includes functions for data preparation, as this is a crucial step for the training process that, in a pipeline of this nature, can't be done during the preprocessing stage. The reasoning behind this decision is to allow the same starting point for all models, without limiting their capabilities to further improve the quality of the training data. This can be done by the inclusion of operations such as feature encoding, treatment of missing values, standardization of values, among others.

Additionally, this phase needs to take into account the learning method of different models when training them. The first step is to define base values for the partitioning of the dataset. Despite common processes suggesting datasets to be split into two partitions, train and test, to ensure fairness and simulate real scenarios, the dataset should be split into the following data-frames:

- Offline Training - This is data that all the models will have access to when training. When available, the data points should be labeled as to allow supervised techniques.
- Online Training - This data will be used to simulate the learning of online models after their deployment. The training is

Feature	Data Type	Description
ip.len	float64	Length of the IP packet in bytes
ip.ttl	float64	Time to Live: Number of hops the packet can take before being discarded
tcp.srcport	float64	TCP Source Port
tcp.dstport	float64	TCP Destination Port
udp.srcport	float64	UDP Source Port
udp.dstport	float64	UDP Destionation Port
tcp.len	float64	Length of the TCP segment in bytes
udp.length	float64	Length of the UDP packet in bytes
tcp.flags_syn	bool	Indicates if the SYN flag is set
tcp.flags_ack	bool	Indicates if the ACK flag is set
tcp.flags_fin	bool	Indicates if the FIN flag is set
size	float64	Total size of the packet in bytes
frame_number	float64	Unique identifier for the packet/frame in the capture

Table 4: Selected Features

realized without the presence of labels, independently of the paradigm, as to reflect real scenarios.

- Testing - The test data, similarly to the offline training, should be labeled in order to evaluate the performance of all the models. This, while it doesn't take part in the training process, impacts the partitioning of the dataset and, for that reason, should be taken into account.

After the partitioning is defined, all the models are trained on the available offline data-frame. For offline models, the process of training ends here and they are ready to be submitted for testing, needing to be retrained if new data is presented. On the other hand, models that have the capability to continuously learn from new data, can both be submitted for evaluation (since the model is already ready to classify new data) or, in this case, continue the learning process by artificially simulating a period of online learning. This is done by feeding the online training data to the models for classification (not training), the models will then classify the data as *normal* or *anomaly* and, continuously, adapt their predictions to fit new patterns or insights.

With both training methods completed, all models are then ready to be evaluated against one another. According to the obtained results, it is natural that this process might need to be repeated multiple times to account for insights learned during the comparison of the results. This can include changes on the definition of the training splits, choice of features or even application of different data preparation techniques. For this reasons, having a well defined and efficient process to train all the different models proves to be highly valuable. This values are represented both in their performance and, perhaps more crucially, in assuring that changes in their training process (for example, the percentage of training data) don't affect/benefit a single model, as all models will be retrained to account for it.

5.5 Model Evaluation

To evaluate the performance of the models, a custom evaluator was implemented. Based on the metrics commonly used in the literature, we selected accuracy (the ratio of correctly classified instances to the total instances), precision (the ratio of correctly identified anomalies to the total identified anomalies), recall (the ratio of correctly identified anomalies to the total actual anomalies), and F1-score (the balance between precision and recall) as our primary evaluation criteria. These metrics collectively ensure that the framework can guide users in selecting models that align with specific operational requirements.

Model	Learning	Accuracy	Precision	Recall	F1 Score
Random Forest	Offline	0.968	0.976	0.956	0.966
Decision Tree	Offline	0.856	0.958	0.729	0.828
SVM	Offline	0.968	0.936	0.999	0.967
SGD (Hinge)	Online	0.470	0.471	0.988	0.638
SGD (Log Loss)	Online	0.472	0.474	0.981	0.641
Bagging	Online	0.893	0.814	1.000	0.897

Table 5: Model Performance

This comparison evaluates the performance of different models trained using either offline or online learning approaches. For the basic training conducted, the results highlight the strengths and trade-offs among the methods tested.

- Random Forest exhibited a strong balance across all metrics (Accuracy: 96.8%, F1 Score: 96.6%), reflecting its robustness and reliability in detecting anomalies with minimal false positives;
- Decision Tree, while simpler, achieved lower recall (72.9%), indicating a tendency to miss some anomalies, yet maintained a high precision (95.8%), making it suitable for scenarios prioritizing false alarm reduction;
- SVM demonstrated near-perfect recall (99.9%), ensuring almost no anomalies were missed, but slightly lower precision (93.6%) indicates a higher false alarm rate. This trade-off may be acceptable in environments requiring maximum anomaly detection;
- Stochastic Gradient Descent (SGD) with Hinge and Log Loss showed modest accuracy (47%), but exceptionally high recall (98.8%), emphasizing their potential for detecting anomalies in real-time. However, the low precision (47.4%) highlights challenges with false alarms, which the framework must address through tuning or combining models;

- Bagging emerged as a highly effective online approach, achieving a strong balance of accuracy (89.3%) and F1 Score (89.7%), with perfect recall (100%). This makes it particularly well-suited for real-time applications where missing an anomaly could have severe consequences.

Additionally, many studies also utilize the ROC curve, which was included in this work to provide a comprehensive analysis of the models' performance [22] [40].

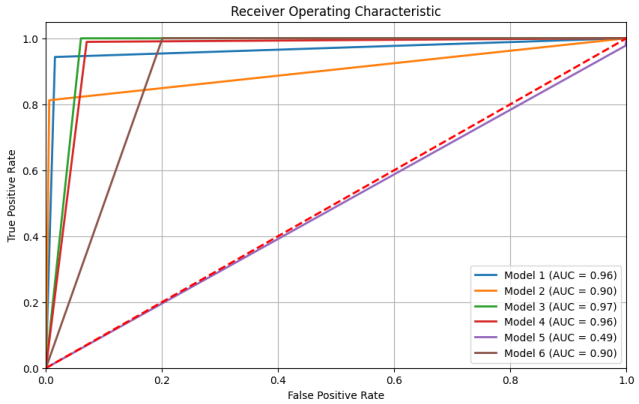

Figure 4: ROC Curve

Another way to visualize the results is through MLflow. We added a connector to MLflow that allows users to upload and run their models seamlessly. Users can choose whether to integrate their work with MLflow or keep it independent. This approach offers flexibility, providing both the option of a local, server-free visualization and the advanced features of MLflow integration.

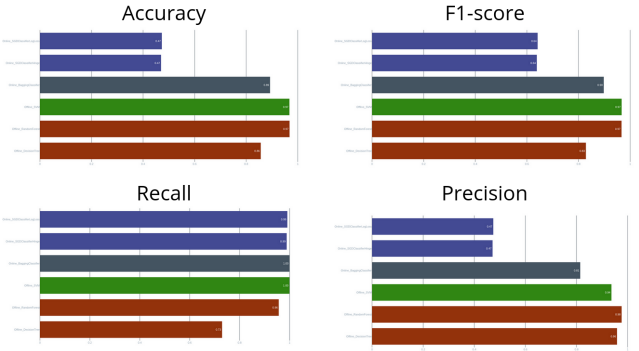

Figure 5: MLFlow Result Visualization

To evaluate the efficiency of the structured pipeline, a time analysis was conducted to determine its impact on reducing the time required to obtain model results. The results are presented in Table 6. The Data Preprocessing stage (1551.39s), Getting Training Data (1.74s), Validation step (5.29s) and Evaluating the prediction (4.11s), which remain constant across all models, have been excluded from the table for clarity.

Without the pipeline, every execution of a model would require running all steps, resulting in a total runtime exceeding 1559 seconds. However, by leveraging the pipeline, independent tasks can be executed separately and the models themselves are able to be saved/retrieved remotely, reducing the need for retraining of preexisting models. This optimization allows the model-specific training phase to take, in this case and with these models, less than 9 seconds per model, significantly improving efficiency. This is further relevant as models can all be trained using the same cached data, which is crucial for comparatively evaluating the various models.

Model	Training	Online Training	Prediction	Model Total Time
OfflineRF	0.10	-	0.14	0.24
OfflineDT	0.01	-	0.09	0.10
OfflineSVM	0.06	-	0.34	0.40
OnlineSGDHinge	0.01	0.85	0.06	0.92
OnlineSGDLogLoss	0.01	0.93	0.10	1.04
OnlineBagging	0.68	0.76	5.25	6.69

Table 6: Execution times in seconds

To ensure fair and consistent comparisons between models, the framework incorporates a mechanism to get the data from the dataset based on a predefined seed. By using the same seed, the framework guarantees that the data remains identical across different runs. This reproducibility not only enhances the reliability of the evaluation process but also provides users with the confidence that model performance differences arise solely from the models themselves, not variations in the data.

Furthermore, during the evaluation process, the proposed pipeline allows to test more complex scenarios, a method not previously found during the research of current literature. For example, models might undergo, when in real scenarios, moments where only anomalous network traffic exists. In order to test this case, we simulated this conditions and repeated the previous evaluations steps, obtaining the following results:

Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.272	1.00	0.272	0.428
Decision Tree	0.181	1.00	0.181	0.307
SVM	0.149	1.00	0.149	0.259
SGD (Hinge)	0.990	1.00	0.990	0.995
SGD (Log Loss)	0.995	1.00	0.995	0.997
Bagging	1.00	1.00	1.00	1.00

Table 7: Model Performance when evaluation data composed only of anomalies

This test, while not providing definitive evidence, shows strong trends to the adaptability of Online Models to the environment presented. This can further consolidate their ability to react and promptly take action on emerging threats, something not displayed by their counterparts. Moreover, even though the model labeled as "Bagging" presents perfect scores across the board, it should be carefully investigated and tested, accounting for concepts such as over-fitting to the data and concept drift. All of these variables are very important to the construction of solid models that can be confidently deployed to real scenarios, proving once more the need for a well structured and adaptable pipeline.

Summing up, the current evaluation process provides the ability to:

- Validate models and their adaptability to different network scenarios;
- Highlight trade-offs in performance metrics that are critical for anomaly detection;
- Provide a clear basis for users to choose models based on operational and functional needs (e.g., precision-focused for stable networks or recall-focused for high-risk environments).

5.6 Model Deployment

The final step of the pipeline is the deployment stage, where models take on the crucial role of classifying live network traffic. This stage includes selecting the most appropriate models for deployment based on their evaluated performance and implementing robust monitoring mechanisms, especially for online learning models, to ensure their performance consistently meets predefined standards over time.

Building on the comparisons drawn in the evaluation phase, the pipeline allows us to identify the most suitable models for specific deployment scenarios. Offline-trained models may be the optimal choice for environments where high accuracy on consistent traffic patterns is essential. On the other hand, online learning models are better suited for ever changing environments with evolving network traffic patterns, enabling them to adapt to new and emerging threats (such as zero-days). Analogous to traditional firewall practices, offline models act as predefined rule sets to address known threats, while online models try to improve dynamically. However, this adaptability introduces the risk of model drift, which can compromise the overall security posture if left unchecked.

To mitigate such risks, the deployment stage leverages the capabilities of MLflow and the proposed structured pipeline to ensure models are regularly monitored and evaluated even after being deployed. This continuous evaluation enables early detection of performance degradation, allowing for timely interventions. Since MLFlow saves the state of the models after both training and testing, previous versions can easily be restored to a moment where their performance meets the standards for classifying live traffic. This also applies to offline models as, in order to adapt to changes in traffic patterns, they need to be retrained from zero with new data. As it is expected, the process is not straightforward and its success is not guaranteed, thus being able to reliably test and restore previous versions of models proves useful even in the process of deployment.

This structured and adaptive approach ensures that models remain effective and resilient in real-world scenarios, aligning with the broader goals of maintaining high levels of security and system reliability.

6 Pipeline Evaluation

The proposed pipeline represents a commendable effort to address significant challenges in network traffic classification using machine learning. Its structured approach to reproducibility, fairness amongst models and adaptability sets a strong foundation for advancing in the field of network traffic analysis.

A key strength of the pipeline is its emphasis on reproducibility. By defining distinct, replicable phases—such as data gathering, preprocessing, model training, and evaluation—the framework ensures that future researchers and stakeholders can validate findings and build upon its methodology. The incorporation of a seed-based randomization process for data splitting is particularly valuable, as it guarantees fairness by providing consistent datasets for training and testing across different models. This approach not only enhances reliability but also directly addresses issues of comparability often seen in machine learning research.

Another notable strength of the proposed pipeline is its adaptability to diverse scenarios, accommodating both online and offline learning paradigms, something not previously accounted for. This dual capability bridges a critical gap in the domain, enabling models to handle real-time anomaly detection and retrospective analyses equally well across different learning methodologies seamlessly. Furthermore, the use of well-established tools such as Pyshark for packet processing and MLflow for model tracking adds robustness to the pipeline. These tools ensure accessibility for researchers while maintaining flexibility for customization. The ability to integrate different data formats for training, such as PCAP and CSV, further enhances the pipeline's versatility, tackling the problem of data availability by making it suitable for a wide range of data sources and research goals.

The evaluation methodology employed in the pipeline is comprehensive, with metrics such as accuracy, precision, recall, and F1-score providing a solid basis for assessing model performance. Additionally, the inclusion of ROC curves adds depth to the analysis, but the evaluation could be further enriched by incorporating metrics tailored to real-world applications, such as latency, throughput, and computational efficiency. These metrics are particularly important in network traffic analysis, where real-time detection and classification are critical. While not directly associated with network traffic, incorporating these additional metrics would provide a more technical view of the pipeline's performance and ensure its suitability for deployment in diverse operational scenarios is rigorously tested. This would allow researchers and practitioners to not only compare the classification accuracy of various models but also evaluate their trade-offs in terms of speed and resource demands, ensuring they can meet the stringent requirements of real-world network security applications.

Perhaps the biggest challenge right now is its accessibility for the end user. Since this is a technical proposition for a new pipeline it still might pose a challenge for users less familiar with both machine learning and network security concepts, even though the tools and technologies chosen are widely used among the community. This, while not necessarily negative, as it is expected to be utilized by other researchers and stakeholders in the field, could drive more unexperienced users to fallback to more AD-HOC, and less structured, practices. To combat this, while it's not a direct solution, we created a public repository containing the complete proof of concept, allowing users, experienced and other wise, to further understand and improve on the proposed methodology.

Overall, the proposed pipeline effectively addresses key challenges in network traffic classification while setting the groundwork for future advancements. By balancing reproducibility, adaptability, and user-centric design, it demonstrates the potential to become

a foundational tool for researchers and practitioners alike. With further refinements in accessibility and evaluation, the pipeline can bridge the gap between academic innovation and practical implementation, driving significant progress in machine learning-based network security.

7 Conclusions and future work

In summary, our pipeline demonstrates a thoughtful and well-structured approach to addressing key challenges in machine learning for network traffic analysis. Its strengths in reproducibility, fairness, and adaptability position it as a valuable contribution to the field. The distinction between online and offline learning is particularly noteworthy, as it bridges a critical gap often overlooked in previous studies. By enabling fair comparisons between these paradigms and integrating them seamlessly within the same framework, the pipeline provides valuable insights into their respective strengths and limitations. This capability positions it as a versatile tool for addressing diverse network security challenges, empowering stakeholders to make informed decisions about the best approach for their specific scenarios.

Future work will aim to improve the pipeline by adding real-world metrics like latency, throughput, and resource efficiency to better evaluate its performance in practical settings. These metrics are essential for real-time detection and ensuring the pipeline is suitable for operational use. Additionally, efforts will focus on making the pipeline more accessible by providing clear documentation, easy-to-follow tutorials, and ready-to-use templates. This will help both experienced researchers and less technical users to adopt the pipeline effectively, ensuring its broader impact in network traffic analysis and security applications.

By building on the foundation established in this study, the pipeline has the potential to set a new standard in machine learning for network traffic analysis, fostering reproducibility, innovation, and practical impact. It is our hope that this work will serve as a catalyst for more structured and collaborative efforts in the domain, ultimately advancing the state of network security research and its real-world implementation.

References

[1] [n. d.]. *Canadian Institute for Cybersecurity.* https://www.unb.ca/cic/datasets/
[2] [n. d.]. *KDD CUP 1999: Computer Network Intrusion Detection.* https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data
[3] [n. d.]. *NSL-KDD Dataset.* https://www.unb.ca/cic/datasets/nsl.html
[4] 1998. *Darpa Intrusion Detection Evaluation Dataset.* https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset/
[5] Kayode S Adewole, Taofeekat T Salau-Ibrahim, Agbotiname Lucky Imoize, Idowu Dauda Oladipo, Muyideen AbdulRaheem, Joseph Bamidele Awotunde, Abdullateef O Balogun, Rafiu Mope Isiaka, and Taye Oladele Aro. 2022. Empirical analysis of data streaming and batch learning models for network intrusion detection. *Electronics* 11, 19 (2022), 3109.
[6] Ahmed Abdelmoamen Ahmed and Gbenga Agunsoye. 2021. A real-time network traffic classifier for online applications using machine learning. *Algorithms* 14, 8 (2021), 250.
[7] Hayder Naser Khraibet AL-Behadili. 2021. Decision tree for multiclass classification of firewall access. *International Journal of Intelligent Engineering and Systems* 14, 3 (2021), 294–302.
[8] Qasem Abu Al-Haijaa and Abdelraouf Ishtaiwia. 2021. Machine learning based model to identify firewall decisions to improve cyber-defense. *International Journal on Advanced Science, Engineering and Information Technology* 11, 4 (2021), 1688–1695.
[9] Shumon Alam, Yasin Alam, Suxia Cui, and Cajetan Akujuobi. 2023. Data-driven network analysis for anomaly traffic detection. *Sensors* 23, 19 (2023), 8174.
[10] Azam Andalib and Seyed Morteza Babamir. 2023. Anomaly detection of policies in distributed firewalls using data log analysis. *The Journal of Supercomputing* 79, 17 (2023), 19473–19514.
[11] Jimena Andrade-Hoz, Jose M Alcaraz-Calero, and Qi Wang. 2024. Multi-layer multi-technology firewall optimisation in beyond 5G networks using machine learning classifiers. In *2024 14th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*. IEEE, 563–568.
[12] Ridhwan Athief, Naveen Kishore, R Nithya Paranthaman, et al. 2024. Web Application Firewall Using Machine Learning. In *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. IEEE, 1–7.
[13] Gustavo Betarte, Eduardo Giménez, Rodrigo Martínez, and Alvaro Pardo. 2018. Improving web application firewalls through anomaly detection. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 779–784.
[14] Gustavo Betarte, Eduardo Giménez, Rodrigo Martínez, and Álvaro Pardo. 2018. Machine learning-assisted virtual patching of web applications. *arXiv preprint arXiv:1803.05529* (2018).
[15] Qimin Cao, Yinrong Qiao, and Zhong Lyu. 2017. Machine learning to detect anomalies in web log analysis. In *2017 3rd IEEE international conference on computer and communications (ICCC)*. IEEE, 519–523.
[16] Bahzad Charbuty and Adnan Abdulazeez. 2021. Classification based on decision tree algorithm for machine learning. *Journal of applied science and technology trends* 2, 01 (2021), 20–28.
[17] Padraig Cunningham and Sarah Jane Delany. 2021. K-nearest neighbour classifiers-a tutorial. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–25.
[18] Zhangxuan Dang, Yu Zheng, Xinglin Lin, Chunlei Peng, Qiuyu Chen, and Xinbo Gao. 2024. Semi-Supervised Learning for Anomaly Traffic Detection via Bidirectional Normalizing Flows. *arXiv preprint arXiv:2403.10550* (2024).
[19] C Mitchell Dayton. 1992. Logistic regression analysis. *Stat* 474 (1992), 574.
[20] Fatih Ertam and Mustafa Kaya. 2018. Classification of firewall log files with multiclass support vector machine. In *2018 6th International symposium on digital forensic and security (ISDFS)*. IEEE, 1–4.
[21] Fatih Ertam and Mustafa Kaya. 2018. Classification of firewall log files with multiclass support vector machine. In *2018 6th International symposium on digital forensic and security (ISDFS)*. IEEE, 1–4.
[22] Cyril Goutte and Eric Gaussier. 2005. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *European conference on information retrieval*. Springer, 345–359.
[23] Vikramaditya Jakkula. 2006. Tutorial on support vector machine (svm). *School of EECS, Washington State University* 37, 2.5 (2006), 3.
[24] Adrian Komadina, Ivan Kovačević, Bruno Štengl, and Stjepan Groš. 2024. Comparative Analysis of Anomaly Detection Approaches in Firewall Logs: Integrating Light-Weight Synthesis of Security Logs and Artificially Generated Attack Detection. *Sensors* 24, 8 (2024), 2636.
[25] Kahraman Kostas. 2018. Anomaly detection in networks using machine learning. *Research Proposal* 23 (2018), 343.
[26] Yafei Li and Liejun Wang. 2022. Routing network traffic predict based on firewall logs using Machine Learning. In *Proceedings of the 2022 4th International Conference on Robotics, Intelligent Control and Artificial Intelligence*. 833–840.
[27] Yanli Liu, Yourong Wang, and Jian Zhang. 2012. New machine learning algorithm: Random forest. In *Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3*. Springer, 246–252.
[28] Claudio Marques, Silvestre Malta, and João Magalhães. 2021. DNS firewall based on machine learning. *Future Internet* 13, 12 (2021), 309.
[29] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*. IEEE, 1–6.
[30] Punam Mulak and Nitin Talhar. 2015. Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. *Int. J. Sci. Res* 4, 7 (2015), 2319–7064.
[31] Martha Ozohu Musa and Temitope Victor-Ime. 2023. Improving Internet Firewall Using Machine Learning Techniques. *American Journal of Computer Science and Technology* 6, 4 (2023), 170–179.
[32] Adrian Pekar and Richard Jozsa. 2024. Evaluating ML-Based Anomaly Detection Across Datasets of Varied Integrity: A Case Study. *arXiv preprint arXiv:2401.16843* (2024).
[33] Vinícius F Santos, Célio Albuquerque, Diego Passos, Silvio E Quincozes, and Daniel Mossé. 2023. Assessing machine learning techniques for intrusion detection in cyber-physical systems. *Energies* 16, 16 (2023), 6058.
[34] Aref Shaheed and MHD Bassam Kurdy. 2022. Web application firewall using machine learning and features engineering. *Security and Communication Networks* 2022, 1 (2022), 5280158.
[35] Amin Shahraki, Mahmoud Abbasi, Amir Taherkordi, and Anca Delia Jurcut. 2022. A comparative study on online machine learning techniques for network traffic streams analysis. *Computer Networks* 207 (2022), 108836.

[36] Deepanshu Sharma, Vansh Wason, and Prashant Johri. 2021. Optimized classification of firewall log data using heterogeneous ensemble techniques. In *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 368–372.
[37] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. 2017. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.
[38] Erdem Ucar and Erkan Ozhan. 2017. The analysis of firewall policy through machine learning and data mining. *Wireless Personal Communications* 96 (2017), 2891–2909.
[39] Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. 2018. An anomaly detection method to detect web attacks using stacked auto-encoder. In *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE, 131–134.
[40] Ž Vujović et al. 2021. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications* 12, 6 (2021), 599–606.
[41] Harry Zhang. 2004. The optimality of naive Bayes. *Aa* 1, 2 (2004), 3.

A Glossary

This section provides explanations of key terms mentioned in this article to enhance understanding and clarity.

- **Supervised Learning**: A machine learning technique where the learning process relies on labeled data samples. The algorithm uses labeled data during training to learn patterns and relationships, which are later tested for accuracy and effectiveness.
- **Unsupervised Learning**: A machine learning technique that does not rely on labeled data. Instead, the algorithm groups data based on shared characteristics, analyzing relationships between these groups.
- **Reinforcement Learning**: A machine learning method where the algorithm learns through a system of rewards and penalties. Correct decisions are rewarded, while incorrect ones are penalized, leading to refined behavior over time.
- **Concept Drift**: The phenomenon in machine learning where the statistical properties of the target variable or data distribution change over time. This can degrade the performance of models trained on past data, requiring retraining or adaptation to maintain accuracy.
- **TP (True Positive)**: Correctly identified attack data (classified as an attack).
- **FP (False Positive)**: Benign data incorrectly classified as an attack (Type-1 Error).
- **FN (False Negative)**: Attack data incorrectly classified as benign (Type-2 Error).
- **TN (True Negative)**: Correctly identified benign data (classified as benign).
- **Accuracy**: The proportion of correctly classified data (both attacks and benign) out of the total dataset.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

- **Recall**: The proportion of correctly identified attack data to the total actual attack data.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Precision**: The proportion of correctly identified attack data to all data classified as attacks.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **F-measure**: The harmonic mean of precision and recall, providing a balanced metric when dealing with imbalanced data.

$$\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC Curve (Receiver Operating Characteristic)**: A graphical representation of the classifier's performance across different thresholds, plotting the True Positive Rate (Recall) against the False Positive Rate.
- **Denial-of-Service (DoS)**: A cyberattack where the attacker floods a target system or network with excessive requests, rendering it unavailable to legitimate users.
- **Phishing**: A social engineering attack where cybercriminals impersonate trustworthy entities (e.g., via email or websites) to deceive individuals into revealing sensitive information.
- **Zero-Day**: A security vulnerability in software or hardware that is unknown to the vendor or the public. Attackers can exploit it before it is patched.

B Frameworks

In this section, we present the frameworks referenced throughout the paper, as discussed in the section on current limitations, with the captions indicating the source papers.

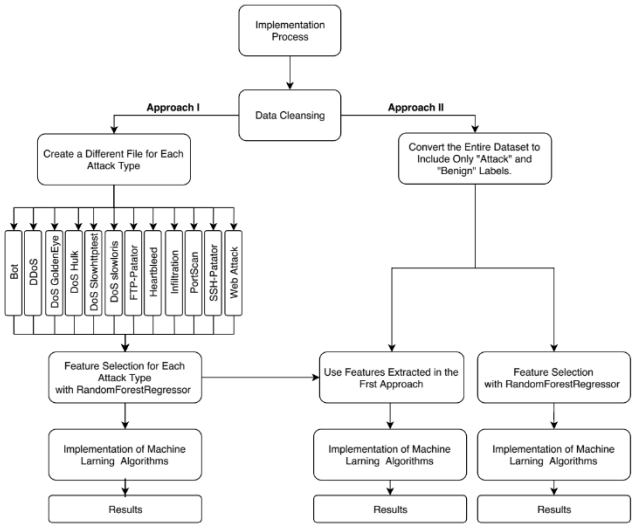

Figure 6: Anomaly Detection in Networks Using Machine Learning Framework

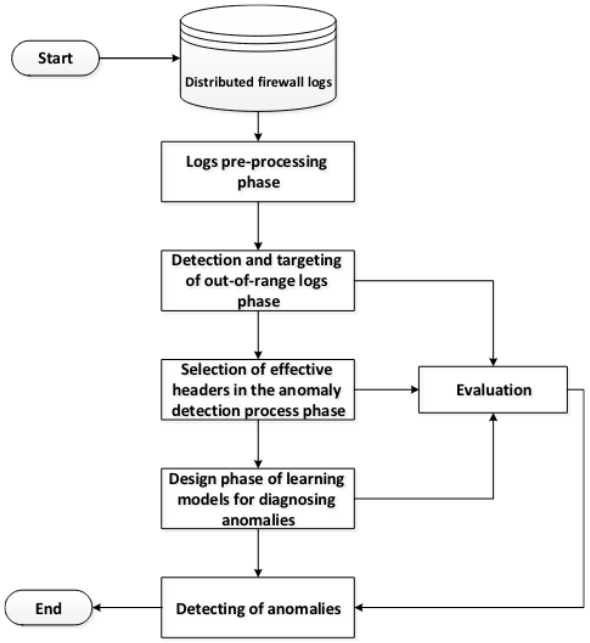

Figure 7: Anomaly detection of policies in distributed firewalls using data log analysis

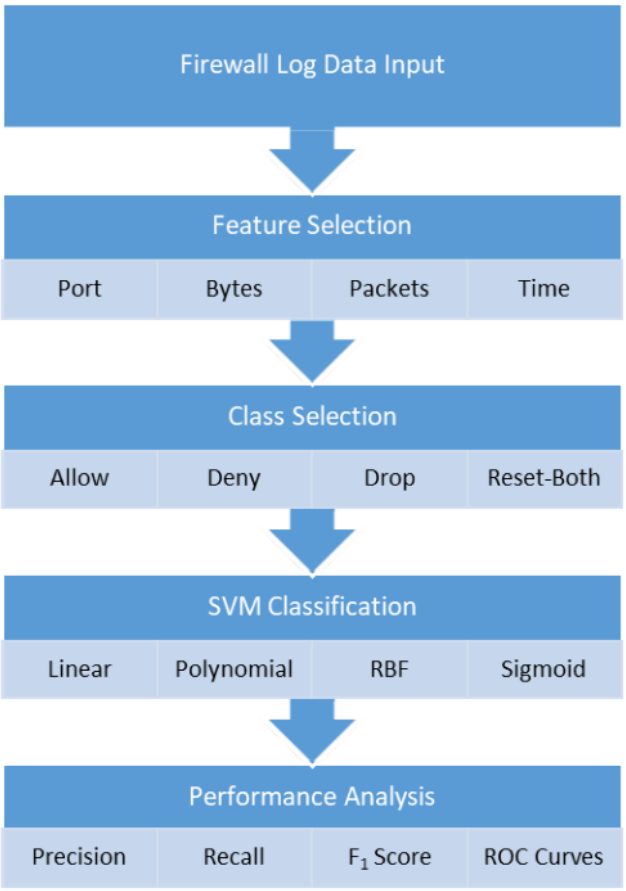

Figure 8: Classification of Firewall Log Files with Multiclass Support Vector Machine

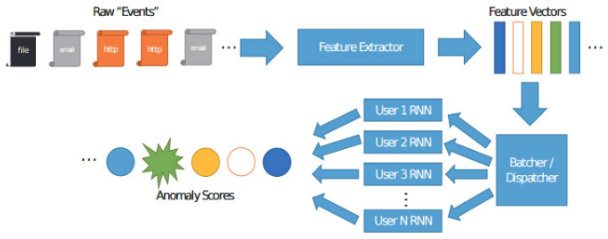

Figure 9: Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams

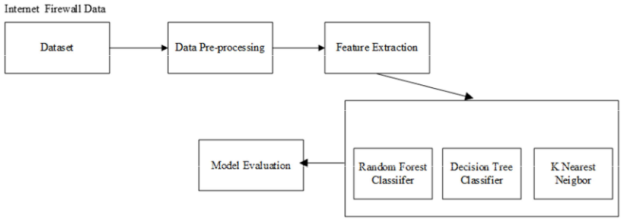

Figure 10: Improving Internet Firewall Using Machine Learning Techniques

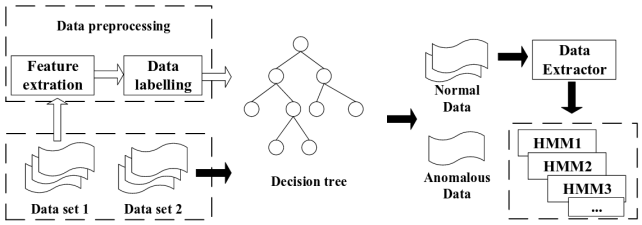

Figure 11: Machine Learning to Detect Anomalies in Web Log Analysis

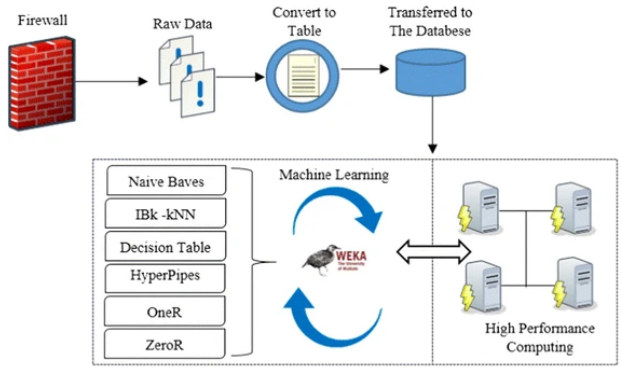

Figure 12: The Analysis of Firewall Policy Through Machine Learning and Data Mining

Figure 13: Web Application Firewall Using Machine Learning