Enhancing Network Security through Machine Learning: A Study on Intrusion Detection Systems

Filippo Ciliberti
Universita' di Bologna - Campus di Cesena
Digital Transformation Management
filippo.ciliberti@studio.unibo.it
Cesena, 04/10/2024

***Abstract*—This research aims to analyze the possibility to apply different machine learning approaches in order to enhance the effectiveness and reliability of the network intrusion detection. To check the performance of the proposed approach with conventional methods, it is applied to detect both normal and anomalous flows extracted from a real military network environment, a data set obtained from kaggle. Upon importing the dataset, the distribution and nature of the features where examined through an exploratory data analysis (EDA) phase. Visualizing the dataset helped me understand better how to elaborate on the project, as well as handling outliers, or creating new attributes. A variety of supervised learning techniques were explored and evaluated. The outcomes of this study would be used in proposing an Intrusion Detection System that is effective and can be used to improve cybersecurity in the organizational networks.**

1. Introduction

The world is more interconnected than ever. Today's organizations are exposed to a constantly growing number of network based threats that jeopardize the confidentiality, integrity and availability of their assets.

To cope with threats and menaces, SIEM systems record logs and events, so as to report any kind of illegal activity within a network. These information come from Intrusion Detection Systems that are software based application that detect anomalous and illegitimate behaviors, trigger alerts, allow for fast recovery and increase the cybersecurity resilience of organizations.

Nowadays, threats are enhanced in terms of complexity and adopt very sophisticated techniques. Traditional security measures are therefore outdated, as well as IDS.

In this research I explored my datasets before to improve my knowledge and address what I though was the most appropriate elaboration according the goal. Data visualization helped me in identifying patterns and trends and managing any data missing or null values to keep data consistency. I also discussed categorical elements that, while critical, had numerous zero values, which might disrupt the model's learning process. I found and kept outliers since they usually reflect anomalies important for intrusion detection.

I applied both feature engineering and label encoding to finetune the dataset. Converting categorical features into numeric values for machine learning processing became especially necessary due to label encoding. I then devised a correlation matrix to study the relationships between the remaining multiple features, helping me to avoid duplicate details in my model.

When the dataset was ready for use after cleaning and optimization, I split 80% for training and 20% for testing.

Finally, this research is based on the application of several machine learning approaches to improve IDS efficiency and effectiveness. Supervised learning, is very promising in enhancing the surveillance of network intrusions by analyzing patterns and estimate both normal or anomalous traffic, therefore threats.

The training and testing of multiple machine learning techniques—including Logistic Regression, Support Vector Machines, Decision Trees, Random Forest, AdaBoost, and Neural Networks—were part of the modeling for the research, designed for the classification of network connections.

To determine how effectively they can discover anomalies, these models were reviewed against a test dataset. The purpose of this study is to identify the best models to enhance the detection performance of IDS and boost organizational network defenses.

A key objective of the research is to find the ideal machine learning techniques for execution, while also comparing different traffic patterns and characteristics that could indicate potential and specific attacks.

2. Related Work

Understanding the goals of my project and subjectively understanding the kind of data I owned led me to start researching related works and datasets found on the internet. I discovered that the large majority of publicly available datasets categorize network traffic according to advance defined attack types, frequently labeled in the dataset itself. This style was not in tune with the path I planned to travel. As a result, I selected to persevere with my own strategy and to greatly understand the features within my dataset.

An especially useful resource in this process was the KDD Cup 1999 dataset which, even though it was different from mine, resembled it in many aspects. The dataset presented in KDD Cup 1999 helped me to clarify the definitions of a variety of attributes that appeared in both datasets. This phase was key to fully understanding the core meaning of the features and their relation to multiple traits of network traffic.

Given this advanced insight, I focused to the effective understanding and labeling of the features, with the goal of preparing the data correctly for supervised machine learning. While concentrating on the features themselves instead of using standard labels for attack types, I endeavored to develop a model that could discern unusual network behavior from the data relationships, making my method more flexible and suitable for diverse scenarios.

3. Proposed Method

Traditionally, IDS have been categorized in signatures and anomaly detection based. Signature-based intrusion detection systems checks new traffic against established attack patterns (signatures) which are constantly updated against a database of signatures, whereas anomaly-based systems identify uniqueness in behavior by comparing it ideal behaviors. While anomaly-based systems are more versatile, they still deal with the problem of high false positive rates. This was also took in consideration in choosing the best model for this system. Indeed, the goal is to create an anomaly based system that minimize the false positives but offers a robust approach to identify normal or anomalous traffic, so as to make IDS stronger.

The training of models on datasets enables them to identify difference between normal and abnormal behavior, including in dynamic and difficult network ecosystems. Using this strategy reduces our dependence on manual rule definition and allows us to identify fresh and changing threats.

Having understood the category in which my IDS falls into, I began my project.

3.1. Choice of Dataset

The choice of my data, was driven by the need of gaining useful datasets on the publicy available databases. On Kaggle, I found the ”Network Intrusion Detection” dataset that provides analysis on a simulated LAN which constitutes an emulation of the military network environment of the US Air Force.

The dataset is composed of ”Train.csv” which contain several connections to the network towards the understanding of weather is is “normal” or an ”anomalous” connection.

3.2. Importing the needed libraries

To perform data analysis and in developing the model and to evaluate the model performance I required the following libraries. “Pandas” was employed in data manipulation and cleaning since the features found in the dataset are structured. “Matplotlib” and “seaborn” enabled me to visually explain the distribution and the trends of the data set and make sense of data. To handle data in an efficient manner particularly arrays and data processing the ‘Numpy’ supported a large number of mathematical operations.

Scikit learn coded and evaluated models of logistic regression, decision tree classification, support vector classification, random forest, Ada boost and multi-layer perceptron classifier for completing work in classification.

Due to my evaluation, I was able to gauge the performance of the models in addition to determining their accuracy rate, precision, and recall; additionally, I identified their confusion matrix to determine the efficiency of the models.

3.3. Exploratory Data Analysis

The dataset applied in this study has 25,192 entries and 42 features, each highlighting several aspects of network traffic. Such features encompasses a combination of numerical and categorical data, starting from protocol types and service flags to source and destination byte counts, and also includes the number of access ’logged-in’ ’error rates.

The train.info() outlined the data structure, inclusive of column names, data types, and any missing values. All columns in the dataset are complete, with no missing entries, and consist of three primary data types: int64, float64, and object. Data must be encoded before applying any sort of machine learning.

Further exploratory investigation was done using train.describe(include=’object’). I was able to yield descriptive statistics for the categorical features, understanding the different protocol types, services, and connection flags, that populated the dataset. The network traffic types were various.

Aside from that, features including protocol-type, service, and flag allow for insights into the kind of network connections being investigated, and numerical features like ’src-bytes’ and ’dst-bytes’ provide important information about data transfer.

This first investigation demonstrated that the dataset is properly set up, featuring no absent values and a suitable combination of numerical and categorical information. The multiplicity of connection-specific features, along with error rates and traffic types, produces a rich framework for crafting machine learning models to identify anomalies and malicious behavior in the network.

3.4. Understanding Attributes

Crucial was to understand the attributes of the data, and how they could have weighted differently on the classification of the network traffic. The table, attached to the ”ipynb” file, extensively explains each feature and its role in the network capture.

In addition, for what concerns the attribute ”flag”, there is a ”Flag Code Table” in the same document which describes each flag and its role in the connection.

3.5. Visualizing data

The attributes of the dataset were investigated within the exploratory analysis to understand network traffic distribution. An important step in this was to analyze connection flags, protocols, services, and the fraction of anomalies in the data.

The flags identified a mixture of network activities, and the lion’s share, representing the 59.4% of the data, were marked as ’SF’ (normal establishment and termination). Important additional flags included ’S0’ (27.82 %), showing failed connections, and ’REJ’ (8.80 per cent), which represents invalidated connections. Using less common flags, namely ’RSTR’, ’SH’, and ’RSTO’, together contributed smaller fractions, reflecting a variety of incomplete or abnormal connections. This flag analysis provides important understanding of the blend of conventional and irregular network actions present in the dataset. I have grouped the flags below a ”rare$_{t}hreshold”of0.5intoa”rare_{f}lag”attribute.$

When we analyze protocol usage, TCP dominated, making up 81.48 % of network traffic, and UDP accounted for 11.95 % and ICMP comprised 6.57 %. For several network environments, the extensive dependence on TCP is regular, pointing out its function in administering reliable, connection-oriented traffic. The availability of UDP and ICMP traffic points to the variety of communication protocols throughout the network.

The services incorporated in the network showed further patterns. The 31.77 % of the traffic was attributed to HTTP, which demonstrates elevated web activity. The services that were critical included (17.27 %) private along with domain-u (7.22 %), SMTP (5.75 %), and FTP data (5.54 %), demonstrating various standard and private protocols, especially in file transfer and email services. I have grouped the services below a threshold of 2% into a ”rare$_{s}ervice”attribute.$

Finally, the dataset presented a large number of anomalies, as 46.61 % of connections were deemed anomalous. The large fraction of irregular activity points out the important demand for effective intrusion detection systems that can spot and answer to possible security dangers.

This study gives a thorough perspective on the composition of network traffic, which sets the stage for the machine learning models that will serve to reveal intrusions.

where X indicates the feature value, μ represents the feature’s mean, and σ its standard deviation.

Z-score measures the statistical distance between a data point and the mean of a dataset. It’s expressed in terms of standard deviations. Usually, a Z-score exceeding 3 signals reveals an outlier.

After applying this rule, I spotted a large number of outliers throughout various features. In any case, I decided against removing or taking care of these outliers when preprocessing my data. In many cases, outliers reveal abnormal behavior in the network, which could be critical for finding anomalies or irregular patterns vital for intrusion detection. Due to the critical focus on detecting anomalous activity in this study, I decided to keep these outliers, as they could be important to identifying potential security threats.

3.6. Feature Engineering

The principal aim during my feature engineering was to improve the dataset by removing less relevant features for the analysis. Identification of features with many zero entries possibly indicated either redundancy or a limitation in their utility for classification. Sorting these features according to the %age of zeros revealed which columns might provide less to the overall model performance.

As a result, I have included all the feature in a list, considered significant for my models. In this ”feature-list” I have included features such as connection flags, error rates, types of service, together with other network properties, which were considered critical for the identification of network patterns.

The list of important features, once finalized, was benchmarked against the entire suite of available features in the dataset. Features not part of this refined list were the candidates to be removed. The excluded columns usually led to a high %age of zero values or were assessed to be of less importance for detecting abnormal behavior.

By ”pruning” the dataset it became optimized for modeling and boosted efficiency by eliminating unnecessary noise and complexity.

3.7. Attack Table Creation

In the feature selection process, I created subsets of the attributes I found more relevant, in order to distinguish three distinct attack types: DoS/SYN Flood, Data Exfiltration, and Port Scanning. For these attacks, the most suitable features aligned with their associated network traffic behavior. My approach provided the flexibility to designs the feature set for various attack types, which improves the labeling efficiency and boosts the accuracy of the training models.

As an illustration, the selection of features such as ’serror-rate’, ’srv-serror-rate’, and ’same-srv-rate’ came about due to their direct relevance to connection errors and service request patterns characteristic of DoS/SYN Flood attacks. Also, Data Exfiltration attacks associated with the features service, ’dst-bytes’, and ’srv-count’ highlight data transfer behavior and possibly indicate unusual data movement. In the end, ’protocol-type’, count, and ’rerror-rate’ were chosen as features in Port Scanning to monitor the organized probing signatures typical of this kind of attack.

The intent was to improve the dataset by associating each attack type with its most important characteristics to enhance the computer learning model training stage.

3.8. Label Encoding

In the preprocessing stage of the dataset, various transformations were applied to numerical and categorical attributes to ensure that the data was in a suitable format for machine learning models.

Numerical Feature Transformation To standardize the numerical features, only columns containing integer or floating-point values were selected. A log1p transformation was applied to these columns, which computes the natural logarithm of (1 + x). This transformation helps in handling skewed data and ensures that extreme values do not dominate the learning process.

Following this, the numerical attributes were scaled using StandardScaler, which standardizes the data by removing the mean and scaling it to unit variance. This step ensures that all numerical features contribute equally to the model without any dominating due to differing scales.

Class Encoding To facilitate classification, the target variable was encoded into a binary format. Specifically, the class label was transformed such that observations labeled as 'anomaly' were assigned a value of 1, while all other classes were assigned a value of 0. The original class column was then dropped to retain only the encoded version.

Categorical Feature Encoding Categorical attributes, including service, flag, and protocol_type, were transformed using one-hot encoding. This encoding method converts categorical variables into a series of binary variables, allowing machine learning algorithms to process them effectively. One-hot encoding was applied consistently across the dataset to maintain compatibility.

These encoding steps ensure that the dataset is numerically structured while preserving the underlying information necessary for accurate classification. The resulting transformed dataset is now prepared for further model training and evaluation.

3.9. Feature Selection

To address the high dimensionality of the dataset, Mutual Information (MI) feature selection was employed. MI measures the statistical dependency between variables, allowing for the identification of features that contribute the most information relative to the target variable. This approach was chosen to reduce redundancy while preserving the most relevant features for classification.

Following this, a filter-based feature selection was applied using a correlation matrix, excluding the target variable. Highly correlated features, with a correlation coefficient greater than 0.9, were removed. Subsequently, the correlation of each remaining feature with the target was computed, and features exhibiting a correlation coefficient below 0.3 were discarded, ensuring that only features with meaningful relationships to the target were retained.

After feature selection, PCA was applied to further reduce dimensionality. However, upon evaluation, it was determined that critical features such as 'src bytes', 'serror rate', 'duration', and 'rerror rate' should have been retained due to their significance in the intrusion detection system. Additionally, an analysis of the skewness of the remaining features revealed that they were positively skewed, indicating potential preprocessing adjustments to improve model performance.

3.10. Handling Outliers

To enhance data quality and mitigate the influence of extreme values, outlier detection and removal were performed using the Interquartile Range (IQR) method. This technique identifies outliers based on statistical dispersion, ensuring that the dataset retains meaningful patterns while minimizing noise.

Initially, all numerical features (both integer and floating-point data types) were identified for analysis. The first quartile (Q1, 25th percentile) and third quartile (Q3, 75th percentile) were computed for each numerical feature. The interquartile range (IQR) was then derived as the difference between Q3 and Q1, representing the middle 50% of the data distribution.

To define outlier boundaries, the lower bound was set at Q1 - 1.5 × IQR, and the upper bound at Q3 + 1.5 × IQR. Any data points falling outside this range were considered outliers. The number of outliers detected per feature was recorded, highlighting that the most affected features were 'src bytes' (66 outliers), 'duration' (2,024 outliers), and 'rerror rate' (3,207 outliers), while other features remained largely unaffected.

Following the identification of outliers, rows containing at least one outlier were removed from the dataset. This resulted in a reduction of the dataset size from 25,192 samples to 20,332 samples, eliminating approximately 19% of the original data. The removal of extreme values ensures a more robust dataset, improving model generalization by reducing the impact of anomalies.

This preprocessing step is critical in intrusion detection systems, where outliers may represent rare yet significant events. Therefore, while the method enhances data consistency, further evaluation is necessary to ensure that essential attack patterns are not inadvertently removed.

3.11. Hold Out, 80 train, 20 test

The first step was to split the dataset into train and test datasets. The Stratified Split method played a crucial role in protecting the class distribution (normal versus anomalous connections) in both the training and testing sets. In imbalanced datasets specifically, this is an important technique, since it confirms that the model acquires knowledge from a balanced set of data.

There was a split in the dataset of 80% for training and 20% for testing.

4. Results

4.1. Model Training and Cross-Validation

To assess the performance of different machine learning models I used cross-validation since it strongly estimates how effectively a model will perform by splitting the training dataset into a number of subsets known as 'folds.' Specifically, I used 5-fold cross-validation, which works as follows: the training data set is broken apart into 5 equal sections. Through all iterations, a specific subset acts as the validation set and the model receives training from the additional 4 subsets. This project completes its course 5 times, where each subset acts as a validation set only one time. The final score regarding accuracy is determined after all iterations are performed. It was essential to reduce overfitting and ensuring that the model performs well across different sections of data, instead of a singles test split.

I trained six different models and for each of them I assessed the accuracy.

- **Logistic Regression Cross-Validation Accuracy**: 0.9018
- **Decision Tree Cross-Validation Accuracy**: 0.9964
-
- **SVM Cross-Validation Accuracy**:0.9777
- **Random Forest Cross-Validation Accuracy**: 0.9964
- **AdaBoost Cross-Validation Accuracy**: 0.9594
- **ANN Cross-Validation Accuracy**: 0.9824

4.2. Model Evaluation on Test Data

5. Model Evaluation and Performance Analysis

After conducting cross-validation, the models were evaluated on the test set to assess their generalization performance. The evaluation metrics used include *accuracy*, *precision*, *recall*, and the *confusion matrix*, which provides a detailed breakdown of correct and incorrect classifications. The results for each model are presented below.

5.1. Logistic Regression

- **Accuracy:** 0.9098
- **Precision:** 0.9216
- **Recall:** 0.9012

5.2. Decision Tree

- **Accuracy:** 0.9983
- **Precision:** 0.9982
- **Recall:** 0.9983

5.3. Support Vector Machine (SVM)

- **Accuracy:** 0.9761
- **Precision:** 0.9766
- **Recall:** 0.9751

5.4. Random Forest

- **Accuracy:** 0.9978
- **Precision:** 0.9978
- **Recall:** 0.9977

5.5. AdaBoost

- **Accuracy:** 0.9631
- **Precision:** 0.9649
- **Recall:** 0.9607

5.6. Artificial Neural Network (ANN)

- **Accuracy:** 0.9786
- **Precision:** 0.9790
- **Recall:** 0.9777

These results indicate that tree-based models, particularly **Decision Tree** and **Random Forest**, achieved the highest classification performance, with accuracies above 99.7%. However, **Logistic Regression** and **AdaBoost** showed slightly lower performance, with accuracies around 90.9% and 96.3%, respectively. The **SVM and ANN models** performed competitively, with accuracies above 97%, demonstrating their robustness for intrusion detection.

Further analysis is required to determine the trade-offs between model complexity, interpretability, and computational efficiency. Tree-based models exhibited near-perfect classification accuracy, but their performance should be validated against potential overfitting concerns.

Artificial Neural Network - MPL
The Artificial Neural Network (ANN), implemented as a Multilayer Perceptron (MLP), also achieved excellent results:

- **Epochs**: 300
- **Test Accuracy**: 0.9860
- **Loss**: 0.0280

The evaluation of ANN model over 300 epochs, revealed insightful results shown as well in the learning curve and accuracy graphs.

5.7. Model Performance Visualization

5.7.1. Learning Curve. The learning curve shows the progression of training loss and validation loss over 300 epochs. Initially, both losses start at a high value, but they decrease rapidly within the first 50 epochs, indicating that the model is learning effectively. As training progresses, the losses continue to decline and stabilize around epoch 100, with minor fluctuations observed thereafter. The validation loss closely follows the training loss, suggesting that the model generalizes well and does not suffer from overfitting. If the validation loss had increased while the training loss continued to decrease, it would have indicated overfitting. However, since both curves remain closely aligned, the model appears to have achieved a good balance between learning from the training data and performing well on unseen data. Some fluctuations in loss are present, which could be further smoothed out by techniques such as early stopping, learning rate adjustments, or additional regularization. Overall, the learning curve suggests that the model has been trained effectively and does not exhibit signs of severe overfitting or underfitting.

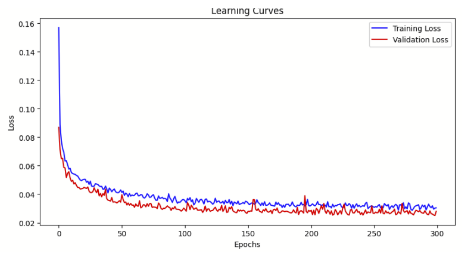
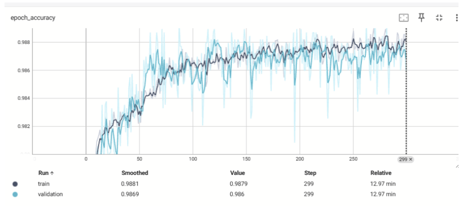


Figure 1. Learning curves

5.7.2. Accuracy Curve. he accuracy curve shows the progression of training and validation accuracy over 299 epochs. Initially, both accuracies start at a lower value and steadily increase, indicating that the model is effectively learning from the data. Around epoch 50, the validation accuracy stabilizes, closely following the training accuracy, demonstrating that the model generalizes well to unseen data. Throughout the training process, minor fluctuations are present, particularly in the validation accuracy, which suggests some variability but remains within an acceptable range. By the final epoch, the training accuracy reaches approximately 0.9879, while the validation accuracy stabilizes at 0.9869, showing a strong performance with minimal overfitting. The close alignment of the two curves suggests that the model maintains a good balance between learning and generalization.



	Run ↑	Smoothed	Value	Step	Relative
●	train	0.9881	0.9879	299	12.97 min
●	validation	0.9869	0.986	299	12.97 min

Figure 2. Accuracy over 300 epochs

6. Findings

These findings provide strong evidence that machine learning models are highly effective for Intrusion Detection Systems (IDS). The high accuracy rates observed, reaching 98.79% for training and 98.69% for validation, demonstrate the robustness and reliability of these models in identifying anomalous network behavior with minimal overfitting. The close alignment between training and validation accuracy further indicates that the models generalize well to unseen data, ensuring stable performance in real-world applications.

For an IDS, maintaining both high precision and recall is crucial to correctly distinguishing normal network traffic from potential threats while keeping the false positive rate low. The minimal discrepancy between training and validation loss suggests that these models effectively avoid misclassifying benign traffic as malicious, which is essential for reducing unnecessary alerts and operational disruptions. At the same time, the high recall values ensure that actual threats are detected with strong reliability, minimizing the risk of missing critical security incidents.

From a practical standpoint, these results suggest that the IDS can leverage these models to continuously monitor network activity in real time and accurately flag suspicious behavior with a high degree of confidence. The strong generalization capabilities observed indicate that these models are not only effective at recognizing known attack patterns but are also adaptable to emerging threats and evolving cyber risks. Consequently, these findings highlight the practical significance of machine learning in enhancing network security and improving the resilience of IDS solutions.

6.1. Conclusion

This research utilized machine learning models to enhance the effectiveness and reliability of Intrusion Detection Systems (IDS) , which are now fundamental to the protection of organizational networks. By implementing a thorough data preprocessing, visualization, and feature engineering pipeline , the objective was to develop a system capable of accurately distinguishing between normal network activity and potential security threats using advanced machine learning techniques.

Among the models tested, Random Forest and Artificial Neural Networks (ANN) emerged as the most suitable candidates for IDS deployment, demonstrating exceptional classification performance . Their ability to capture complex interactions between network traffic features made them particularly well-suited for this application, ensuring both high accuracy and adaptability in detecting cyber threats.

These findings are not merely of theoretical importance but provide practical real-world value for intrusion detection. An effective IDS must not only detect anomalies with high reliability but also minimize false positives to prevent disruptions in network operations. The results confirm that both Random Forest and ANN models meet these critical requirements. Their high precision ensures that benign network traffic is rarely misclassified as a threat, while their strong recall guarantees that actual attacks are not overlooked. This balance is crucial for real-time IDS implementations, where the system must continuously adapt to evolving cybersecurity threats.

The results of this study further establish that machine learning models significantly enhance IDS capabilities by providing a robust, scalable, and efficient solution for modern network security challenges. Their high accuracy, precision, and recall suggest that they are not only ready for deployment in real-world environments but also possess the flexibility to evolve alongside emerging cybersecurity threats. By meeting the fundamental requirements of intrusion detection, these models lay the foundation for the next generation of intelligent IDS solutions, capable of adapting to increasingly sophisticated attack vectors and strengthening network resilience in dynamic environments.

References

[1] Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection: Results from the JAM Project by Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan.

[2] Sampada Bhosale - Network Intrusion Detection

[3] Abhay Mudgal - Intrusion Detection System