

**Threat Intelligence and Incident Response (CYBR 5940)**

**A Report on Intrusion Detection System using Machine Learning Algorithms**

By

**Madhavi Mamidala(700744319)**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Contents** | **Page Number** |
| **1.** | **Introduction** | **3** |
| **2.** | **Dataset** | **5** |
| **3.** | **Setup Process & Toolkit** | **5** |
| **4.** | **Algorithm-Testing** |  |
| **A.** | **Decision Tree Classification** | **6** |
| **B.** | **Support Vector Machine Algorithm** | **7** |
| **C.** | **Neural networks** | **8** |
| **5.** | **Analysis Dataset** | **10** |
| **6.** | **Results** | **14** |
| **7.** | **Conclusion** | **17** |
| **8.** | **References and Bibliography** | **18** |

# **Introduction**

Cybersecurity is not a PANACEA for all cyber-attacks; While it is possible to mitigate the risk of becoming a victim, complete elimination of cyber-attacks is not feasible.

In this report, we will be discussing web attacks and how to detect them using machine learning algorithms.

**What makes us choose this topic?**

In the era computers, Understanding and effectively detecting web attacks is crucial for safeguarding online assets, securing user data, and preserving the integrity of web applications. It enables organizations and individuals to gain insights into attackers' techniques and tools, empowering them to develop effective strategies for mitigating and responding to such threats.

Detecting web attacks involves employing a combination of security measures like intrusion detection systems (IDS), web application firewalls (WAF), log analysis, and threat intelligence. However, it remains an ongoing challenge due to the ever-evolving nature of web attack techniques and the constant emergence of new vulnerabilities.

**How can we minimize the risk of being a victim of web attacks?**

Minimizing web attacks requires a proactive approach and the implementation of various security measures. To minimize the risk of such attacks, organizations and individuals can implement several key strategies like Implement strong network security measures, antivirus and anti-malware software, enable network encryption, monitor network activity and using Machine Learning for detecting the web Attacks.

To further enhance the effectiveness of web attack detection and minimize the risk, organizations and individuals can leverage the power of machine learning. By implementing machine learning algorithms specifically designed for detecting web attacks, they can improve their security posture and proactively identify suspicious or malicious activities.

**How can we implement machine learning to detect web attacks?**

Machine Learning (ML) plays a significant role in enhancing the effectiveness and efficiency of IDS. ML algorithms enable IDS to analyze large volumes of data, identify complex patterns, and adapt to evolving threats in real-time. Here's how ML helps in analyzing the data within an IDS:

* Anomaly Detection: ML algorithms can learn normal patterns of network behavior by analyzing historical data. When new data arrives, the algorithms compare it to the learned patterns and identify any anomalies or deviations. This approach is particularly useful in detecting previously unseen attacks or zero-day vulnerabilities.
* Signature-based Detection: ML techniques can be employed to generate accurate attack signatures by analyzing known attack patterns. These signatures are then used to match against network traffic and identify specific attacks in real-time.
* Behavioral Analysis: ML algorithms can model and learn the behavior of users, applications, and devices within a network. By establishing a baseline of expected behavior, any deviations from the norm can be flagged as potential threats. This approach is effective in detecting insider threats and advanced persistent threats (APTs).
* Real-time Response: ML algorithms can process and analyze network traffic in real-time, allowing for immediate detection and response to threats. ML-powered IDS systems can automatically generate alerts, trigger defensive actions, or initiate incident response processes to mitigate the impact of an intrusion.
* Continuous Learning: ML models can continuously learn and adapt to new attack techniques and patterns. By regularly updating their knowledge with the latest threat intelligence, ML-based IDS systems can stay effective against emerging and evolving threats.

In this project, we are restricting ourselves with Anomaly Detection due to the Time constraint. To get into anomaly detection, it identifies the instances or events that differ significantly from the expected or normal behavior within a given context. In the realm of cybersecurity, it focuses on detecting unusual patterns, activities, or network traffic that may indicate malicious intent or system vulnerabilities. Anomalies can arise from various factors, such as system failures, software bugs, insider attacks, or sophisticated external threats.

**The Role of Machine Learning:**

Machine learning techniques have revolutionized anomaly detection by providing efficient and scalable solutions to process vast amounts of data and detect subtle deviations. ML algorithms excel at learning patterns from historical data and generating models that capture the normal behavior of a system or network. By comparing real-time data against these learned models, anomalies can be identified based on deviations from expected patterns.

There are many Learning methods for implementing Machine Learning; in this project, we have used Supervised Learning.

**Supervised Anomaly Detection:**

In this approach, the ML model is trained using labeled data that represents both normal and anomalous instances. The model learns to classify new instances as normal or anomalous based on the patterns observed during training. Support Vector Machines (SVM), Decision Tree, and Neural Networks are commonly used supervised anomaly detection algorithms.

**Benefits of Anomaly Detection using Machine Learning:**

* **Early Threat Detection**: Anomaly detection enables the identification of potential threats at their nascent stage, allowing security teams to respond promptly and prevent security breaches before they cause significant damage.
* **Reduced False Positives**: ML algorithms can adapt and learn from the data, minimizing false positives by distinguishing between benign anomalies and genuine security threats, thereby reducing unnecessary alarms and alert fatigue.
* **Scalability and Efficienc**y: ML-powered anomaly detection systems can handle large-scale data sets, enabling real-time analysis of network traffic and identification of anomalies in complex and dynamic environments.
* **Adaptability to Changing Patterns**: ML models can continuously learn from new data, adapt to evolving threat landscapes, and update anomaly detection patterns accordingly. This ensures the system remains effective against emerging and sophisticated threats.
* **Improved Security Posture**: By accurately identifying anomalies, ML-driven anomaly detection systems significantly enhance an organization's security posture, providing insights into vulnerabilities and aiding in the development of robust defense mechanisms.

# **Dataset**

**Where did we get the dataset?**

The dataset utilized for our analysis was obtained from the UNB (University of New Brunswick) website. The link is below:

<https://www.unb.ca/cic/datasets/ids-2017.html>

The purpose of uploading this dataset is to gain insights into the individuals or entities involved in Intrusion Web Attacks. The dataset contains various attributes related to the attacks, including *Destination Port, Flow Duration, Total Fwd Packets, Total Backward Packets, Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean* and approximately 70 other columns that comprise the entire dataset. By analyzing these attributes, we aim to uncover patterns and trends that can help identify the characteristics and behaviors of defaulters involved in Intrusion Web Attacks.

**What changes have been made to the existing dataset, with reference?**

To facilitate easy processing by Machine Learning algorithms, the dataset has already been formatted in CSV (Comma-Separated Values) format. To work with a manageable subset of data, the original dataset containing 173,600 records has been reduced to 10,000 records. Within this subset, there are 7,828 records classified as "Benign" and 2,180 records classified as "Web attack." This reduction in data size allows for more efficient analysis and model training while capturing many instances for both types of records.

**Setup Process & Toolkit**

**Software And Hardware Requirements**

* **Operating System: Window 10**
* **IDE: Jupyter Notebook, Anaconda**
* **RAM: 16 Gb**
* **HDD: 1 Tb**
* **Programming Language: Python**
* **Packages: Pandas, Numpy, SKLearn**

Jupyter Notebook was utilized as the preferred development environment for our data science projects.

Python was chosen as the preferred programming language for our data science projects due to its ease of use, effectiveness, and widespread adoption in the field. To facilitate the execution of our code, we utilized popular Python-based tools such as Anaconda, Juypter Notebook which provides a comprehensive environment for writing, running, and executing code.

For implementing various algorithms related to classification, regression, and clustering, specific libraries and toolkits were utilized. Notable among them are Scikit-Learn, Pandas, and NumPy. These libraries offer a range of functionalities and pre-built functions that simplify the implementation of machine learning models and data analysis tasks.

**Decision Tree Classification:**

Decision tree classification is a supervised learning algorithm that uses a tree-like structure to make predictions. The tree is made up of nodes, where each node represents a decision that needs to be made. The leaves of the tree represent the possible outcomes of the decision.

The decision tree is trained by splitting the data into smaller and smaller subsets until each subset is homogeneous. This means all the data points in a subset belong to the same class.

Once the tree is trained, it can be used to make predictions by starting at the root node and following the branches until a leaf node is reached. The class of the leaf node is then the predicted class for the data point.

Diagram

Description automatically generated

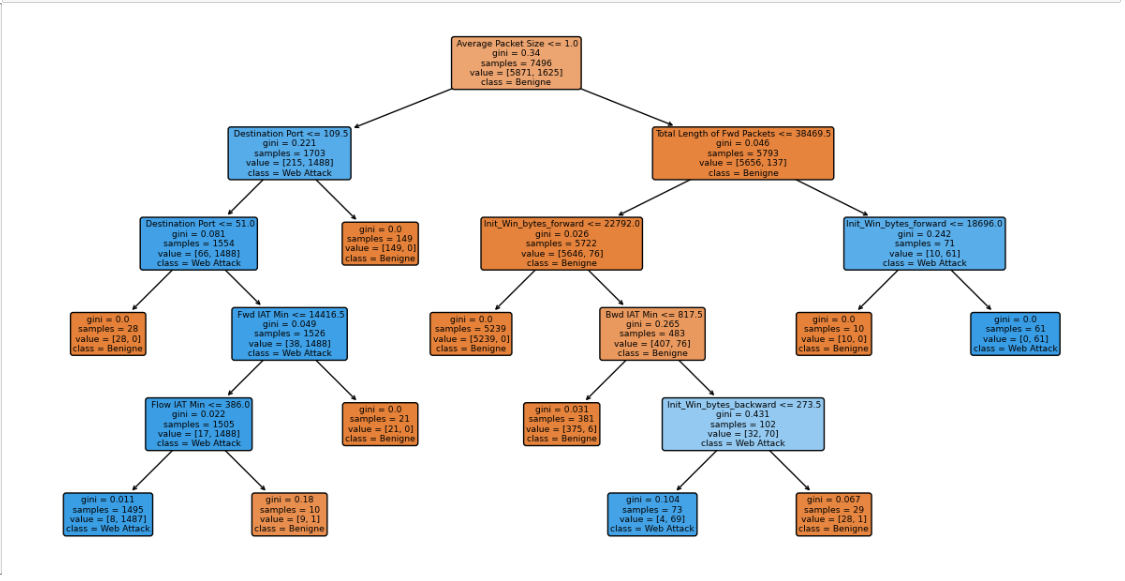
**Source: https://www.researchgate.net/publication/261635169/figure/fig2/AS:669546729971726@1536643788603/A-simplistic-decision-tree-for-the-detection-of-a-user-to-root-attack-on-a-computer.png**

The decision tree's internal nodes represent attribute tests, while the leaf nodes represent the final classification or decision. The algorithm uses a series of if-else conditions based on the attribute values to traverse the tree until reaching a leaf node, which provides the predicted class label.

**Decision Tree Classifier:**

Graphical user interface, text

Description automatically generated with medium confidence



Shape, rectangle

Description automatically generated

**Support Vector Machine Algorithm:**



**Source:** [**https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm**](https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm)

A vector model is a way of representing text or other data as a set of vectors. Each vector represents a single feature of the data, such as a word or a concept. The vectors are then used to represent the data in a way that can be processed by machine learning algorithms.

**Goal**: Support Vector Machines aim to create an optimal decision boundary, or hyperplane, that separates data points into classes. By selecting key support vectors, the algorithm ensures accurate classification of new data points. SVM can handle both linear and non-linear classification problems using different kernel functions and is robust to outliers due to its reliance on support vectors.

SVM classified into two types:

**Linear SVM:**

Linear SVM is designed for linearly separable data, where the classes can be accurately separated by a single straight line or hyperplane. The linear SVM classifier aims to find the optimal hyperplane that maximizes the margin between the two classes. It seeks to achieve the largest separation between the support vectors of different classes in a linearly separable dataset. Linear SVM is computationally efficient and works well when the data can be effectively divided by a linear decision boundary.

**Non-linear SVM:**

Non-linear SVM is used for data that is not linearly separable, meaning a straight line or hyperplane cannot accurately classify the classes. In non-linear SVM, the algorithm utilizes kernel functions to transform the original feature space into a higher-dimensional space where linear separation becomes possible. By mapping the data to a higher-dimensional feature space, non-linear SVM finds a decision boundary that can separate the classes effectively. This allows for more complex decision boundaries, such as curves or non-linear shapes, to classify the data accurately.

**SVM Classifier:**

Text

Description automatically generated

**Neural networks (Multilayer perceptron):**

A neural network is a type of machine learning algorithm that is inspired by the human brain. It is made up of a series of interconnected nodes, or neurons, that are arranged in layers. The input layer receives data, the hidden layers process the data, and the output layer produces a prediction or classification.

Neural networks can be used to solve a wide variety of problems, including image recognition, natural language processing, and speech recognition. They are particularly well-suited for problems that are difficult to solve with traditional machine learning algorithms.

Diagram, schematic

Description automatically generated

Source: <https://www.investopedia.com/terms/n/neuralnetwork.asp>

**ML Classifier:**

Graphical user interface, text, application

Description automatically generated

**Overview of the Code**

**Where did we obtain the code, and if not, how can we build the code?**

There are many codes available on the internet for identifying the Web attack and Benign. But I want to produce a new idea for identifying the web attack by using Ideal Alpha value with Cross Validation technique.

**Ideal Alpha Value:**

The ideal alpha value refers to the threshold or significance level used in anomaly detection algorithms. By selecting an appropriate alpha value, the detection system can strike a balance between identifying genuine attacks and minimizing false positives. This value determines the sensitivity of the detection system and plays a vital role in achieving accurate results.

**Cross-Validation Technique:**

Cross-validation is a robust technique used to assess the performance and generalizability of a web attack detection model. It involves partitioning the available dataset into multiple subsets, typically referred to as folds. The model is trained on a combination of folds and tested on the remaining fold iteratively.

By using this organizations can improve the accuracy and effectiveness of their web attack identification systems. These approaches enable better detection capabilities, reduce false positives, and enhance the overall security posture of web applications.

The in-detail explanation of the code is explained in the Analysis of Dataset.

**The In-Depth Explanation of the Code: Analysis of the Dataset**

**Step 1: Importing all the necessary Libraries**

Importing all the required libraries into one place is indeed considered good practice in Python programming. This approach allows for easy modification and management of the imported packages.

Text

Description automatically generated

**Step 2: Importing Dataset & Data reading:**

To read a dataset in CSV format and display the top 5 rows of the data.

A picture containing text

Description automatically generated

**Step 3: Data frame Structure:**

When we calculated the total no. of non-null values, we got 10008 for each datatype-float, integer & object.

Table

Description automatically generated

**Step 4: Data Preprocess and Memory reduction:**

For this data frame a total of 6.0+ Mb of memory is consumed. Integer and float values are float64(24), int64(54), object (1).

**Step 5: Datatype (Object) Analysis:**

For object datatype, we have 4 object types of values (i.e., 'BENIGN', 'Web Attack Brute Force', 'Web Attack XSS', 'Web Attack Sql Injection').

Graphical user interface

Description automatically generated with medium confidence

We are replacing specific class labels in the y variable. It identifies instances where the class label is "Web Attack Brute Force", "Web Attack XSS", or "Web Attack Sql Injection" and replaces them with the simplified class label "Web Attack". The purpose of this step is likely to consolidate the different types of web attacks into a single category for analysis or modeling purposes.

Graphical user interface, application

Description automatically generated

**Step 6: Cost\_complexity\_pruning\_path for calculation the Alpha**

The code performs cost complexity pruning on a Decision Tree classifier by calculating the pruning path, extracting the alpha values, and creating a series of pruned decision tree classifiers with different alpha values. These pruned classifiers can be used for further evaluation, comparison, or selection based on their performance and complexity.Graphical user interface, text, application

Description automatically generated

Chart, box and whisker chart

Description automatically generated

**Step 7:** **Cross Validation for finding the Best Alpha:**

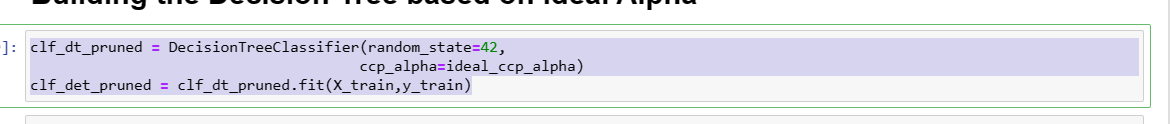
The purpose of performing cross-validation is to obtain a more robust estimate of the model's performance by evaluating it on multiple subsets of the training data. It helps assess how well the model generalizes to unseen data and provides insights into its overall accuracy or performance. The resulting scores can be used to evaluate and compare different models or hyperparameter settings.

Chart, line chart

Description automatically generated

**Step 8: Building the Decision Tree based on Ideal Alpha**

Training a pruned Decision Tree classifier is to find a balance between model complexity and performance. By applying cost complexity pruning, the algorithm prunes away unnecessary branches of the decision tree, reducing overfitting and potentially improving the model's ability to generalize to unseen data. The resulting pruned classifier can be used for prediction or further evaluation on new data.



Timeline

Description automatically generated

**Step 9: Classification using different Machine Learning Algorithms:**

Comparing the training set and testing set with ml model for accuracy. Using MLA\_compare we compare the row index and each set of parameters for every classifier then we got 95% accuracy.

Table

Description automatically generated

# **Results:**

After cleaning, preprocessing, and wrangling the data, evaluating the effectiveness of a machine learning model is crucial to assess its performance. The confusion matrix provides a valuable performance measurement for classification tasks. Here is an explanation of the different elements in the confusion matrix:

* **True Positive (TP):** This represents the number of positive instances correctly predicted as positive by the model. It indicates that the model correctly identified the positive cases.
* **True Negative (TN):** This represents the number of negative instances correctly predicted as negative by the model. It indicates that the model correctly identified the negative cases.
* **False Positive (FP) (Type 1 Error):** This represents the number of negative instances incorrectly predicted as positive by the model. It indicates that the model made a false positive prediction, suggesting the presence of a positive case when it is not.
* **False Negative (FN) (Type 2 Error):** This represents the number of positive instances incorrectly predicted as negative by the model. It indicates that the model made a false negative prediction, failing to detect a positive case.

**Decision Tree Algorithm Results:**

Text

Description automatically generated

Table

Description automatically generated

Text

Description automatically generatedChart, treemap chart

Description automatically generated

After running the data through the Decision Tree algorithm and evaluating the predictions, you found that the accuracy and precision scores of the intrusion dataset are close to or at a good score of 1. This indicates that the algorithm is highly accurate in assessing the data and making predictions**.**

**Support Vector Machine Algorithm Results:**

Table

Description automatically generated

Graphical user interface, text, application

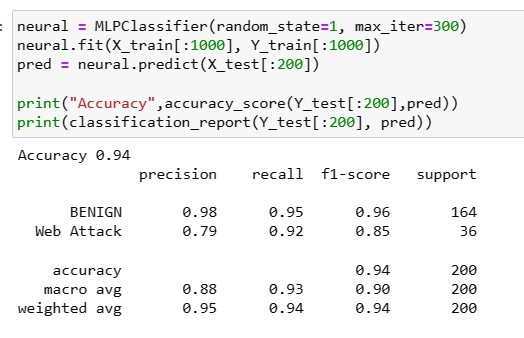
Description automatically generated

Chart, treemap chart

Description automatically generated

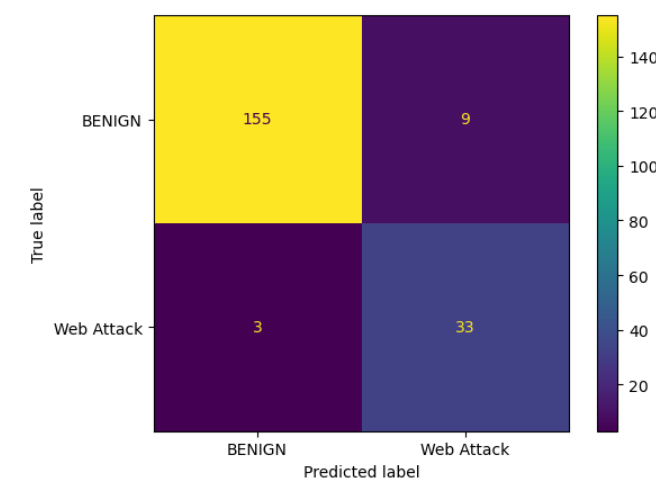
We ran data through the Support Vector Machine and predicted the output and accuracy. The accuracy and precision scores of the intrusion dataset with a difference of 0.6 percent. This means the algorithm is accurate while assessing the data and predicting. This algorithm was accurate assessing both f1 and accuracy.

**Neural networks (Multilayer perceptron):**



A picture containing diagram

Description automatically generated

**\**

We ran data through the Multi-Layer Perceptron and predicted the output and accuracy. The accuracy and precision scores of the credit card dataset with a difference of 0.16 percent. This means the algorithm is accurate while assessing the data and predicting. This algorithm was accurate assessing both f1 and accuracy.

Table

Description automatically generated

# **Conclusion:**

In this study, we employed three different algorithms, namely Decision Tree, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP), to detect and prevent credit card fraud. After evaluating the results, it was found that all three algorithms exhibited high accuracy in detecting fraud cases. However, among these algorithms, the Decision Tree Algorithm demonstrated the best performance, achieving an accuracy score of 0.99, which is close to perfection. Additionally, it exhibited a high F1 score and precision, indicating its ability to accurately identify fraudulent transactions.

Comparing the results presented in the confusion matrix, it is evident that the Decision Tree algorithm outperformed the other algorithms in terms of precision and accuracy when applied to the internet security system dataset. Based on these findings, we recommend utilizing the Decision Tree algorithm as the preferred approach for predicting and detecting intrusion web attacks related to credit card fraud. Its exceptional performance and accuracy make it a reliable choice for ensuring the security of online transactions and protecting users from fraudulent activities.

**Reference:**

1. CART - Classification and Regression Trees, <https://statquest.gumroad.com/l/tzxoh>
2. <https://www.kaggle.com/code/omrastogi/detect-anomaly-cicid-820521>
3. Bhattacharya, S., & Allen, D. (2018). Deep learning for anomaly detection in cyber security. In Proceedings of the ACM Southeast Conference (pp. 1-6).
4. Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. A. (2020). Performance analysis of machine learning algorithms in intrusion detection system: A review. Procedia Computer Science, 171, 1251-1260.
5. Dina, A. S., & Manivannan, D. (2021). Intrusion detection based on machine learning techniques in computer networks. Internet of Things, 16, 100462.
6. Mahfouz, A. M., Venugopal, D., & Shiva, S. G. (2020). Comparative analysis of ML classifiers for network intrusion detection. In Fourth International Congress on Information and Communication Technology: ICICT 2019, London, Volume 2 (pp. 193-207). Springer Singapore.
7. Priyam, A., Abhijeeta, G. R., Rathee, A., & Srivastava, S. (2013). Comparative analysis of decision tree classification algorithms. International Journal of current engineering and technology, 3(2), 334-337.
8. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., & Platt, J. (2001). Support vector method for novelty detection. In Advances in neural information processing systems.
9. Ho, S., Al Jufout, S., Dajani, K., & Mozumdar, M. (2021). A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network. IEEE Open Journal of the Computer Society, 2, 14-25.