CSC 7210: Project 1 Report

Ekle, Ocheme Anthony

October 12, 2023

Abstract

In the past decade, cyber threats have surged. Intrusion detection, a vital for computer security, typically relies on knowledge base rules but struggles with new forms of attacks. We addressed this challenge by developing a machine learning algorithm from scratch, using the decision tree method. Our model attained an impressive 99.99% accuracy in training and 67.7% on unseen data. These results can serve as a valuable benchmark for future research in this field.

1 Introduction

An Intrusion Detection System (IDS) monitors network traffic for suspicious or harmful activities, issuing alerts upon detection [1]. Such intrusions can pose serious threats to security policies, including Confidentiality, Integrity, and Availability (CIA) [2].

Currently, threats to networks and information security remain prominent areas of research and concern. Traditional approaches to intrusion detection often rely on the use of IDS, which depends on a knowledge base containing intricate rules that encapsulate the distinct characteristics of cyberattacks. While these rules form the foundation for intrusion detection, their manual construction is a laborious and time-consuming process.

Traditional IDS frameworks like SNORT [3] focus on signature-based attacks but are limited in handling with emerging cyber threats. Therefore, Data mining and machine learning techniques could provide a alternative solution.

In this data-driven era, machine learning offers a potent solution for advanced cyber attack detection, utilizing Russell and Norvig's Decision Tree algorithm [4]. We harness historical NSL-KDD dataset [5] to distinguish normal network activities from DOS attacks like Neptune, aiming to enhance knowledge representation and evaluate machine learning techniques.

Run source code on terminal using: ***"python Program_source_Code.py"***

2 Method and Model Implementation

2.1 Data Collection and Visualization

The NSL-KDD dataset, sourced from the UNB-Canadian Institute for Cybersecurity [1], comprises 42 columns and **125,973** records in the training set and 2 columns with **22,544** records in the test set.

The dataset encompasses a 'normal' class and 37 unique classes of cyber attacks, including 'neptune', 'warezclient', 'ipsweep', 'portsweep', 'teardrop', 'nmap', and more (see Figure 1). Figure 1 illustrates the distribution of the Top 15 Intrusion Detection Types in the dataset, revealing a significant class imbalance with 53% being normal and 32% being 'neptune'.

2.2 Data Exploration steps

Before inputting the dataset into our machine learning algorithm, I perform the following steps:

1. I load the datasets `KDDTrain.txt` and `KDDTest.txt` using the Python 'pd.read_csv()' function.
2. I rename the 42 columns for both the training and testing datasets using the column names provided in the `KDDTrain.arff` file.
3. After completing the data preprocessing steps, I proceed to build our Decision Tree function.

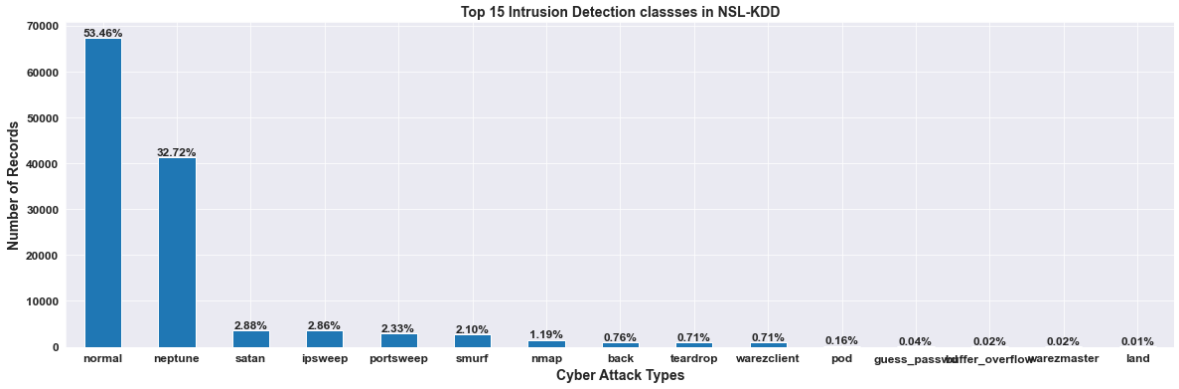

Figure 1: The Distribution of Top 15 intrusion attacks records.

2.3 Possible Feature Engineering Techniques

We could apply the following steps to solve the data imbalance problem, but this is outside the scope of this project.

- Data Augmentation: by creating additional minority classes to balance the distribution.
- Feature Selection to reduce redundant columns, and Resampling methods like SMOTE.
- Convert all text columns to numerical values

2.4 Building the Decision Tree Algorithm

2.4.1 Decision Tree Function

The 'Decision_Tree_Learning' function is defined, which is the core of the Decision Tree Learning algorithm. It is recursive by selecting the best attributes to split the data.

2.4.2 Majority Value

The 'majority_value' function is defined, which calculates the most common class in a set of examples.

2.4.3 Attribute Importance Function

The 'argmax_importance' function is defined, which selects the attribute with the highest importance for splitting the data. Importance is calculated based on information gain.

2.4.4 Entropy Calculation Function

The 'ENTROPY' function is defined, which calculates the entropy of a set of examples, measuring the impurity of the class distribution.

2.4.5 Prediction and Accuracy Functions

- The '**predict**' function is defined to make predictions using the trained decision tree.
- The '**accuracy**' function is defined to calculate the accuracy of the model on the training data.

2.4.6 Testing Data Function

- A function **'train_and_test_decision_tree'** is defined to train and test the decision tree on separate datasets (training and testing).
- The **testing data** is loaded, and accuracy on the **testing data** is calculated and displayed.

```
======================== Training Accuracy ==================
============================================================

Accuracy on training set: 0.9998729886562994

======================== Testing Accuracy ===================
============================================================

Accuracy on testing set: 0.6773864442867282
```

Figure 2: Results: Decision Tree Performance on Training and Testing Data

3 EXPERIMENTS

The experiments were performed on a macOS® Ventura 13.3.1, powered by an Apple® M1 Chip with an 8-Core™ 64-bit processor operating at 3.2 GHz, 7-Core™ GPU, 16-Core™ Neural Engine, 8 GB of unified memory (RAM), and 256 GB SSD storage. The machine learning algorithms were implemented using the Python® programming language and the VsCode™ IDE.

3.1 Results and Discussion

The results of the decision tree model are presented in Figure 2. The model was trained with 123,973 data points and unseen test data of 22544. First, we implemented the decision tree and evaluated the train data with an accuracy of 99.99%. Secondly, we evaluated the unseen data with an accuracy of 67.7%.

Analysis: In summary, the decision tree model demonstrated outstanding accuracy on the training data, showcasing its efficacy in learning from the dataset. Also, it maintained a acceptable accuracy result on the unseen testing data, affirming its practical potential for real-world applications.

4 Conclusion

In Project 1, I successfully implemented a decision tree algorithm with the NSL-KDD dataset. Accuracy metrics were provided for both training and testing phases. The model achieved **99.99**% accuracy in training and **67.7**% on unseen test data, effectively detecting intrusions. These results showcase the model's quality and establish a robust research baseline. **Future work** will address data imbalance and overfitting concerns.

References

[1] TechTarget. What is an intrusion detection system (ids)? definition from searchsecurity. https://www.techtarget.com/searchsecurity/definition/intrusion-detection-system. (Accessed on 10/09/2023).

[2] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.

[3] Abhishek Mitra, Walid Najjar, and Laxmi Bhuyan. Compiling pcre to fpga for accelerating snort ids. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, pages 127–136, 2007.

[4] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[5] UNB. Nsl-kdd — datasets — research — canadian institute for cybersecurity — unb. https://www.unb.ca/cic/datasets/nsl.html. (Accessed on 10/09/2023).