Supervised Machine Learning for DDoS Detection

CPE 400 Final Project

Anna Desorcy

University of Nevada, Reno, CSE Department

May, 2024

Abstract

This project aimed to create a supervised machine learning model to detect various types of DDoS attacks and uncover the most impactful features that determine a malicious or benign attack. This project was created in a Google Colab environment with Python and its associated libraries for data pre-processing, data visualization, and machine learning. Accuracy of detection was tested with multiple supervised models, all achieving over 95% of attacks correctly classified and most impactful features were determined using feature selection methods.

1. Project Overview

Distributed Denial of Service (DDoS) is a large scale cyber attack that attempts to overflow a web server with fake traffic to prevent access from real visitors. This is a common attack used by criminals to disrupt day-to-day traffic, potentially costing a company money or customers. For this final project in Computer Communication Networks, the goal was to design and test machine learning models on a DDoS related dataset to uncover important characteristics of these cyber attacks and to discover how proficient supervised models are on predicting these types of attacks. The dataset chosen for this project was the CIC-DDoS2019 dataset created by the University of New Brunswick (UNB) [9] [10]. This dataset compiles over 400,000 samples, benign and malicious, each with over 70 unique features. Malicious samples were categorized into sixteen different types of DDoS attacks, with the seventeenth category representing a benign sample. The dataset was split into training and testing, with a random sampling of 80% for training and the remaining 20% for testing. The supervised learning models trained on the categories marked malicious, with a ground truth of 1, and those of benign nature, with a ground truth of 0. To determine the features with the highest impact of categorizing samples, a Z-Score Normalization equation was utilized along with a pre-built scoring method designed by the Python sklearn library [8], both of which will be described in detail later on in this report. This paper will further discuss in detail the motivation behind this study, the tech stack, observations, results and analysis, future research and overall takeaways.

2. Motivation

Distributed Denial of Service (DDoS) attacks are one of the most common and simplest types of cyber crimes to perform, making it a prime choice for criminals plotting to disrupt online platforms. Given this information along with knowledge and personal interest of machine learning, this project holds significant real-world value in regards to how these attacks are detected and classified. Using machine learning to detect attacks coupled with the use of fine-tuning will, overtime, lead to higher accuracy models and possible prediction on new or evolving threats.

3. Tech Stack

For this project, Google Colaboratory, a service hosted from Jupyter Notebooks, was used for the development environment. This service offers computing resources, specifically RAM and GPUs, to process data and run machine learning models. Additionally, Python3 has extensive libraries used for data processing and machine learning, hence it was the optimal programming language to choose for this project. It's important to note for this project, a random sampling of To connect the CIC-DDoS2019 dataset to the colab file, the dataset was saved into Google Drive and the folder was mounted into a directory through Google Colab's *drive* library [4]. Table 1 displays all the libraries used in this project. To actually utilize the dataset, a *pandas* DataFrame was chosen to manipulate the data since this data structure works seamlessly with CSV files. To visualize the data, Python's *matplotlib* and *seaborn* was chosen to capture feature selection along with the confusion matrices

Imports
`pandas` [11]
`numpy` [6]
`SelectKBest` [2]
`f_classif` [2]
`train_test_split` [8]
`matplotlib.pyplot` [7]
`DecisionTreeClassifier` [8]
`RandomForestClassifier` [8]
`KNeighborsClassifier` [8]
`Perceptron` [8]
`MLPClassifier` [8]
`accuracy_score` [8]
`confusion_matrix` [8]
`seaborn` [12]

Table 1: Python Libraries

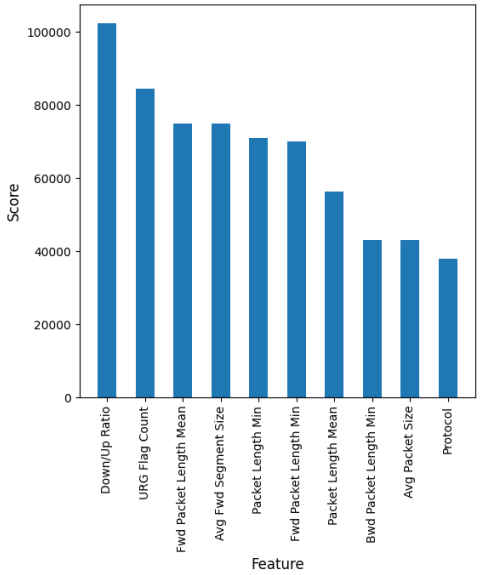


Figure 1: Feature Importance ranked based on scoring algorithm

discussed in the results section. Finally, regarding the supervised learning models and accuracy metrics used for this project, Python's *sklearn* library was used to connect these built in models to the prepared data. Overall, this simple tech stack can adapt to numerous datasets, needing only changes in the data pre-processing to allow the machine learning models to function properly.

4. Observations

While initially examining the CIC-DDoS2019 data set, there was an obvious concern for dimensionality and model explainability for this project. Dimensionality in machine learning is a term used to describe the amount of features a model has to take into consideration when making predictions. A common ailment in machine learning, however, is the curse of dimensionality; the more features a model has, the lower accuracy it could produce. Understanding this complication, the goal was to reduce the amount of features necessary for the model to perform well while systematically uncovering the features that drive its decision making. To do this, the *SelectKBest* [2] function was first used to define the selector used for fitting the training data. Since this dataset involves real values, a high level of numerical variance was present which could cause learning models to find certain features more attractive others. This is a common issue in machine learning that leads to lower accuracy along with inconsistencies in their outputs. To combat this, a technique called normalization is used; in this project, Z-Score Normalization (1) [3] is specifically used. Z-Score Normalization will transform features in a dataset so their distribution equally contributes to the model's predictions. It does this by manipulating the distributions to have a mean of zero and a standard deviation of one, ensuring all the features to be on a uniform scale.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

Once the data was normalized, the selector was fit to the data and features were ranked by their score. The scoring function used in *SelectKBest* was *f_classif* which calculates the ANOVA F-Value, defining the relationship between the feature and its ground truth. If the calculation results in a higher value, this reveals that the feature is heavily dependent on the ground truth, indicating it has a significant impact on how it will be classified. The scores were ranked in descending order and the ten highest ranking features are displayed in 1. Using this selection and scoring process, the number of features was able to be reduced by 87% while still achieving an accuracy of over 95% for all supervised models. These influential features are listed in Table 2 [5].

5. Results and Analysis

While analyzing features that scored highest during feature selection, a few stuck out as the most impactful on classification. The highest scoring feature from the dataset was the ratio of which packets were downloaded and uploaded. The ratio of these speeds could be seen as normal or abnormal, where an abnormal speed might indicate DDoS attack. Since we know DDoS attacks occur by overwhelming a site with requests, an increase in this ratio is an appropriate characteristic, hence

Feature	Description
Down/Up Ratio	Ratio of Download to Upload
URG Flag Count	Number of packets with active URG flag
Fwd Packet Length Mean	Mean length of packet in forward direction
Avg Fwd Segment Size	Average size observed in the forward direction
Packet Length Min	Minimum Length of a Packet
Fwd Packet Length Min	Minimum size of packet in forward direction
Packet Length Mean	Mean length of a packet
Bwd Packet Length Min	Minimum size of packet in backward direction
Avg Packet Size	Average size of packet
Protocol	UDP or TCP Transfer Protocol

Table 2: Highest Scoring Features

an important property in data collection. The values in this feature were binary, meaning the speeds could only be classified as normal or not, making it a great feature for a classification model to learn.

The second highest scoring feature was the URG Flag. A set URG Flag indicates to a system that it needs to prioritize a specific piece of data over others that are being processed. This flag could be incorporated in a DDoS attack since it can be manipulated in a harmful way to make a system prioritize the certain packets associated with the attack, ignoring actual user requests. Similarly to the first feature, this feature also has a binary value, making it, again, important for the supervised model to learn. The majority of the remaining features in the top 10 dealt with numerical values associated with packet length and size, indicating that the actual blueprints of a packet is still highly important to the system.

While inspecting the machine learning models' performances on this dataset, a very impressive accuracy was recorded for each model tested. As seen in Table 3, all models scored higher than a 95% during testing. Most impressively, the decision tree only misclassified 33 samples out of the 40,000 samples tested, as shown in the confusion matrix in Figure 2(a). Regarding the nature of the data itself relating to model performance, the data acts in a somewhat linear way, meaning, classification can be seen as a simple set of rules. This is the way that a decision tree classifies samples; it works through a series of 'questions', making a decision at each step, until a category is found to accurately represent the sample. On the other hand, however, a neural network performs worse in this case because the data follows this set of rules; not allowing the neural network to appropriately use its highly complex classification method. It's also important to note, the perceptron model also had similar accuracy to the neural network, since a perceptron is a single-layered neural network.

Model	Accuracy
Decision Tree	0.9992
KNN	0.9938
Perceptron	0.9573
Neural Network	0.9515

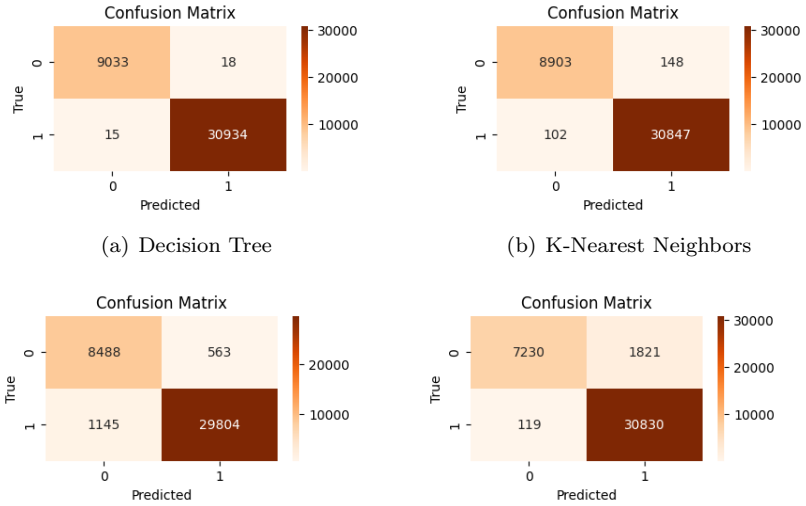
Table 3: Model Performance

The last model tested was K-Nearest Neighbors, a supervised learning model that classifies samples based on the classification of the k closest samples in the sample space. This model also performed extremely well, matching the decision tree's +99% accuracy. Since k is a hyperparameter, the amount of neighbors can be altered to find the highest accuracy. The default value of five neighbors was used during this testing, however, any odd value, three or greater, would be viable for testing if one wished to do so.

Overall, the results of this project aligned with the expectations of high classification performance because the dataset was extremely user-friendly with limited cleaning and processing needed. Given appropriate setup and this type of data, it allows supervised learning algorithms to thrive, gleaning almost all the information possible during training. This program also aligned with the expectation of explainability; features that had the highest impact on the learning models were discovered and aided in the understanding as to why certain classifications were made.

6. Future Research and Takeaways

This project has encouraged me to strive for more than a model with high accuracy, it taught me how to analyze data, look for patterns and the capability to explain the decisions made by the models. Understanding features of network traffic and learning what aspects of them could indi-



(a) Decision Tree

(b) K-Nearest Neighbors

(c) Perceptron

(d) Neural Network

Figure 2: Confusion Matrices for each Machine Learning Model tested

cate a DDoS attack is one of the meaningful takeaways from this project in Computer Communication Networks. The application of machine learning in cybersecurity is a catalyst for understanding and preventing more and more intricate attacks, which has been a growing problem in the field for many years. Growing my skills of data processing, feature selection and machine learning with problems like this is a never ending goal and if I was to further expand this project, I would apply more complex models, techniques or data to find a method to achieve higher accuracy. The source code for this project can be found here [1].

References

[1] Anna Desorcy, *ML_DDoS/CPE400_Project.ipynb*, GitHub, https://github.com/Anna-Desorcy/ML_DDoS/blob/main/CPE400_Project.ipynb, 2024.

[2] K. D, *Optimizing performance: selectkbest for efficient feature selection in machine learning*, (Feb. 2023) https://tinyurl.com/5n7mk6fc.

[3] Google, *Normalization*, Google Developers, https://developers.google.com/machine-learning/data-prep/transform/normalization, 2022.

[4] *Google colaboratory*, https://colab.research.google.com/notebooks/io.ipynb.

[5] A. Habibi Lashkari, G. G. Draper, and M. Mamun, *Github - ahlashkari/cicflowmeter/readme*, GitHub, https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt, 2021.

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy", Nature **585**, 357–362 (2020) 10.1038/s41586-020-2649-2.

[7] J. D. Hunter, "Matplotlib: a 2d graphics environment", Computing in Science & Engineering **9**, 90–95 (2007) 10.1109/MCSE.2007.55.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: machine learning in Python", Journal of Machine Learning Research **12**, 2825–2830 (2011).

[9] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy", 1–8 (2019) 10.1109/CCST.2019.8888419.

[10] M. A. Talukder and M. A. Uddin, "CIC-DDoS2019 Dataset", version V1, 10.17632/ssnc74xm6r.1 (2023) 10.17632/ssnc74xm6r.1.

[11] T. pandas development team, *Pandas-dev/pandas: pandas*, version latest, Feb. 2020, 10.5281/zenodo.3509134.

[12] M. L. Waskom, "Seaborn: statistical data visualization", Journal of Open Source Software **6**, 3021 (2021) 10.21105/joss.03021.