Carter Webb
CPE 400

Final Project Delivery

Project Overview:
This project aimed to analyze network traffic data and predict attacks using machine learning techniques. The dataset is sourced from Kaggle, specifically the "Improved CICIDS2017 and CSECICIDS2018" dataset. I am using a multitude of python libraries including: NumPy, Pandas, Matplotlib, Scikit-Learn, TQDM, imblearn, TensorFlow, and Kaggle.

I decided to work on this project because I am really interested in learning about and applying machine learning techniques in different topics in computer science.

If I had any motivators, I guess it would be the classes that I had this semester. I didn't really know what I wanted to do prior to this semester, but when I started to do work for my Data Mining class, I finally found something that I was really passionate about.

Results and Details:
Some observations I made in this project were as follows:
The minimum forward segment size can be an indicator for a DoS attack. As seen in Figure 1, DoS attacks are especially separated from the Benign (normal) packet transfers.

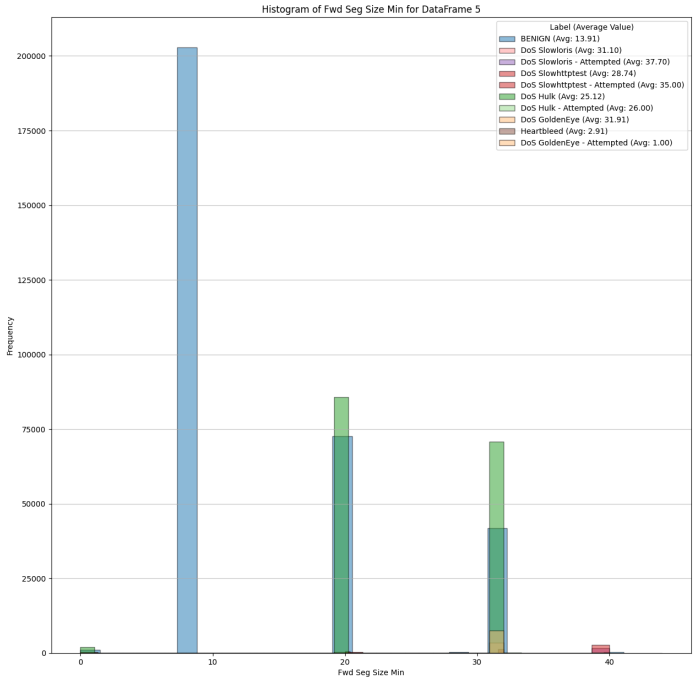

Figure 1. Forward Segment Size Minimum

Another observation that I made is that subflow backward bytes could also be an indicator to DoS attacks. As seen in Figure 2, there is a clear uptick in the amount of bytes transferred and

that seems to correlate with a DoS Hulk attack. This also seems to apply in Figure 3, which shows how the average packet size of a backwards segment aligns with a DoS Hulk attack.

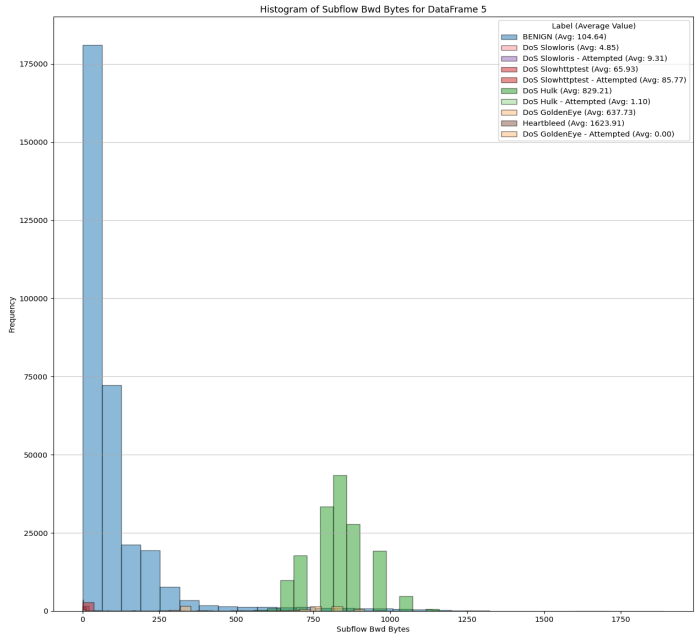

Figure 2. Subflow Backwards Bytes

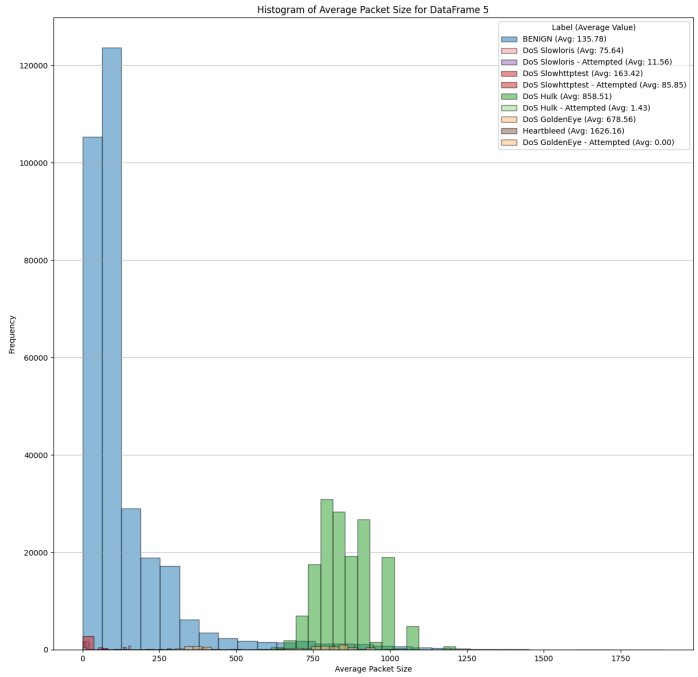

Figure 3. Average Packet Size

Also, as shown in Figure 4 and Figure 5, statistics surrounding average packet length have a heavy indicator of indicating a DoS attack.

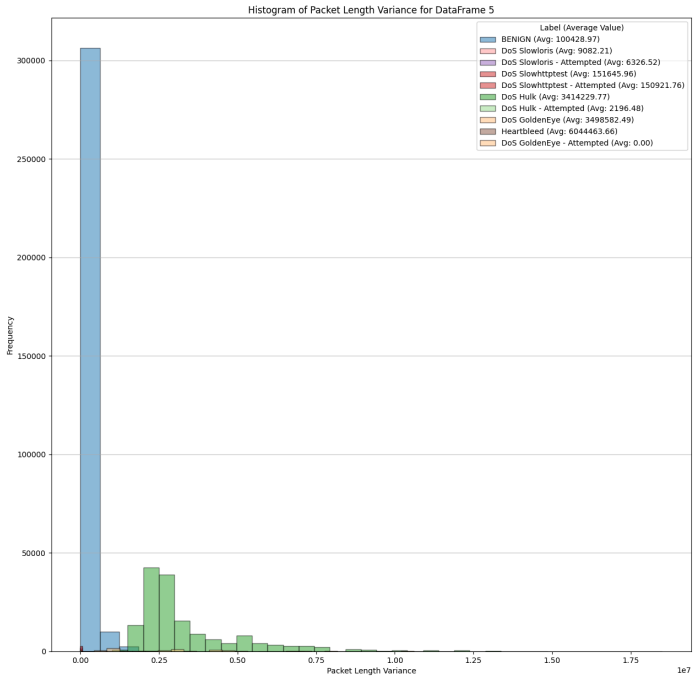

Figure 4. Packet Length Variance

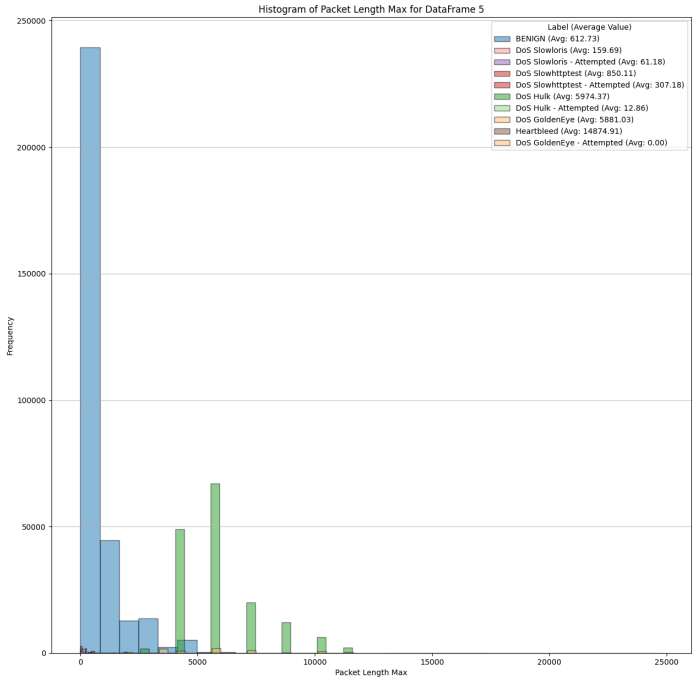

Figure 5. Packet Length Max Distance

Going onto infiltration attacks, it seems that there is a small correlation with the Down/Up ratio of a given packet and some infiltration attacks as shown in Figure 6.

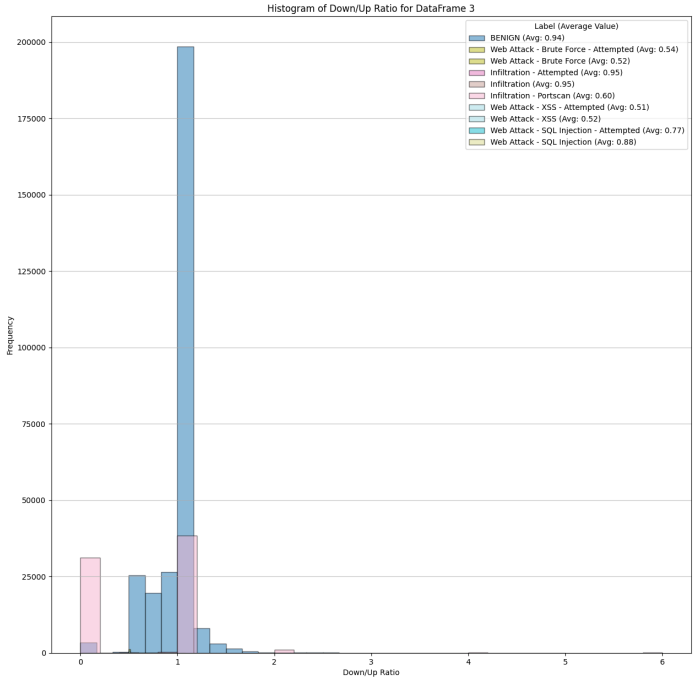

Figure 6. Down/Up Ratio

Web attacks and Infiltration attacks both seem to have a correlation with the flow packet speed as well as the forward packet speed as seen in Figures 7 and 8.

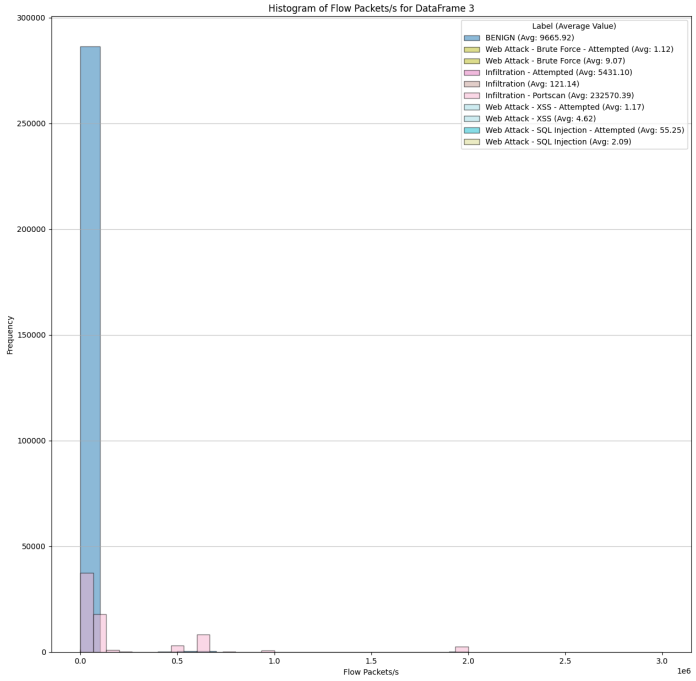

Figure 7. Flow Packets

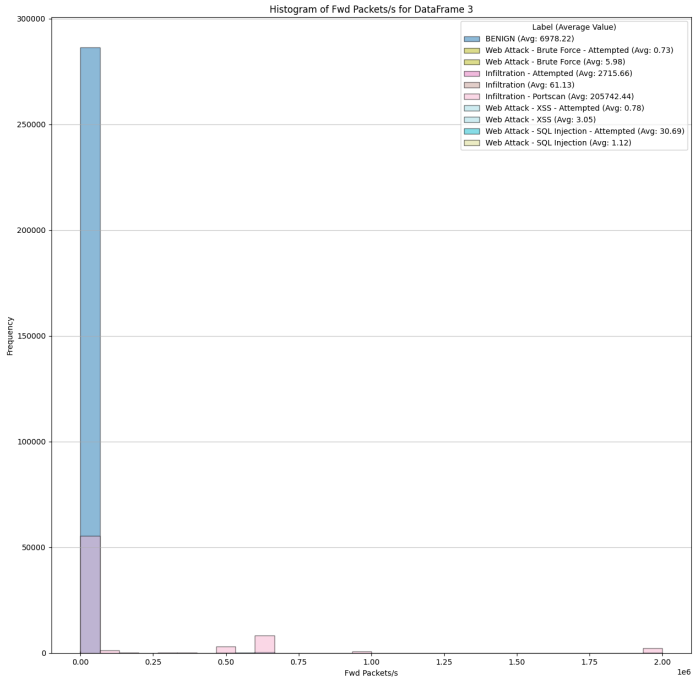

Figure 8. Forwards Packets per Second

In Figure 9, we can see that backwards reset flags have a tendency to be a portscan.

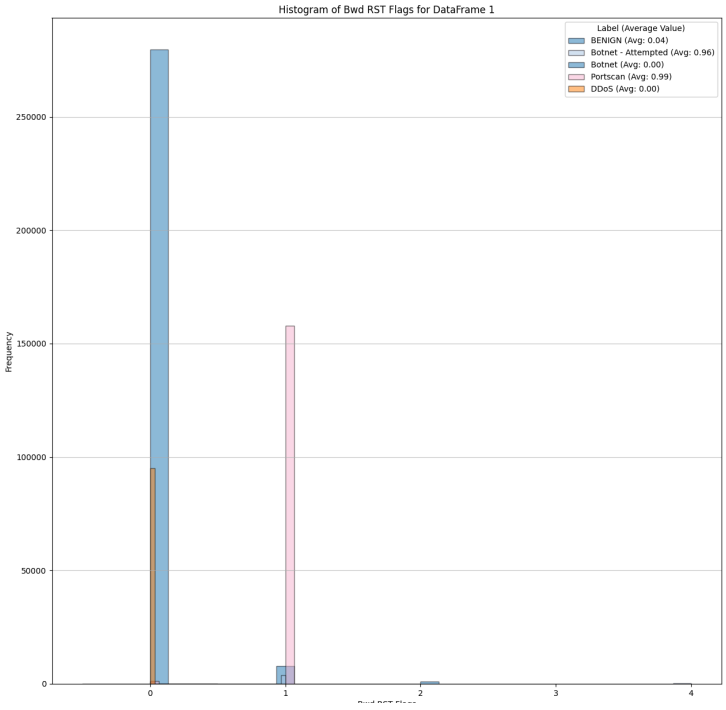

Figure 9. Backwards Reset Flags

The amount of SYN flags that a packet has also seems to help point out whether an attack is occuring or not - specifically for a portscan as seen in Figure 10.

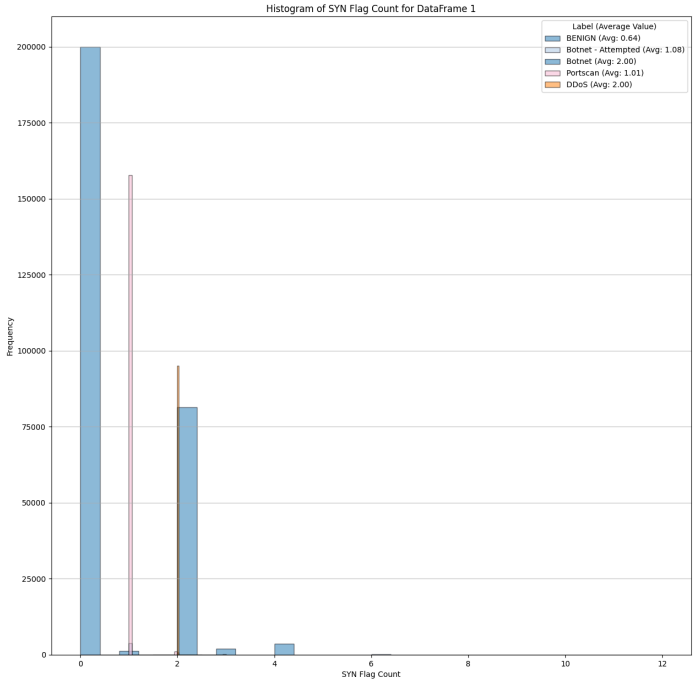

Figure 10. SYN Flag Count

After this I would like to look at and compare the performance of both of the machine learning functions that I used. Using a confusion matrix, we can evaluate the performance of our learning algorithm to see how well it performed. As seen by the first algorithm I used in Figure 11, Naive Bayes was incredibly inaccurate. In a confusion matrix, the ideal pattern we like to see is for a diagonal line down the middle. We can see that it is somewhat there, but we also see that a very large portion of the labels had been incorrectly labeled as a Benign label (label 0), which led to the incredible inaccuracy of the algorithm. Some reasons for this could be the fact that the amount of Benign labels is substantially higher than most of the - in this dataset, Benign labels make up 316418 of the labels out of 419996. The next largest label is label 18, which had 31850 labels. Another reason that there might've been poor accuracy is due to the fact that packets were generally very much like one another.

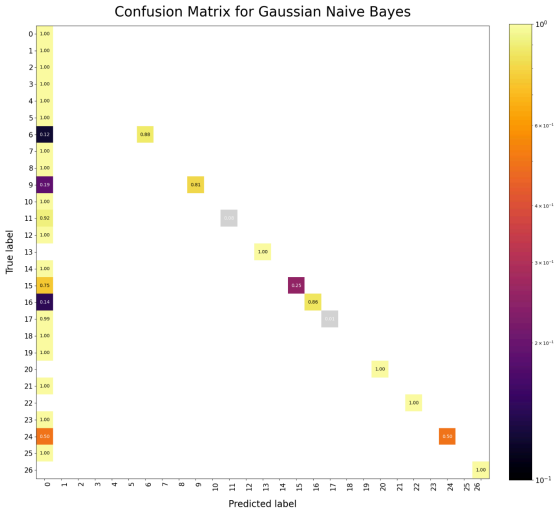

Figure 11. Naive Bayes Confusion Matrix

Now, moving onto the other algorithm I used, we can look at Figure 12 for the Logistic Regression confusion matrix. What is immediately clear is the substantial accuracy - especially when compared to Naive Bayes. Nearly all labels were predicted at 99% accuracy or higher, with there being two main outliers in labels 15 and 16. We can see that the logistic regression algorithm was able to nearly perfectly fit to the dataset leading to great accuracy.

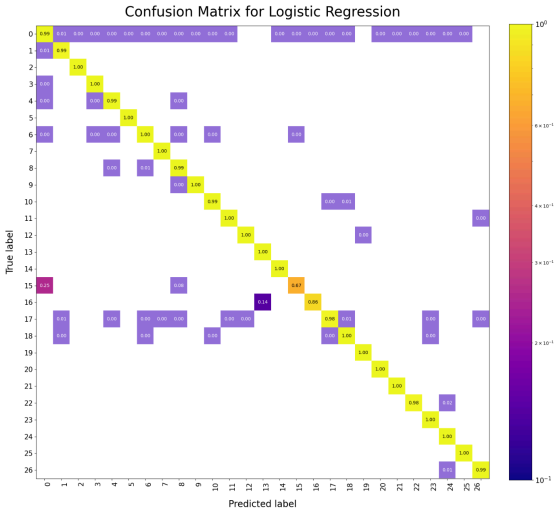

Figure 12. Logistic Regression Confusion Matrix

Now, going into detail about the tech stack that I used, there are 4 main things I would like to note on.

First is the programming language. In this project, I used Python. I decided to do this due to its simplicity and the robust libraries available with regards to machine learning.

Next, I would like to note on the machine learning and data processing libraries which I used. In this project, I used scikit-learn and NumPy. From scikit, I used the two algorithms I talked about earlier - Naive Bayes and Logistic Regression. Scikit helped me train my models and evaluate performance. I used NumPy for numerical operations, data handling, and reshaping arrays. NumPy helps ensure the data was in the correct format before using it.

After that, I would like to talk about the visualization part of my project. Matplotlib helped me make and show the graphs and confusion matrices as you see above.

Lastly, I would like to talk about my development environment. This project originally started in Google Colab, but since the datasets were so large, I decided to move it to my personal computer. I kept the Colab functionality, but made it an optional question at the start of running the program. When coding on my personal machine, I used Jupyter Notebook, which is almost exactly like Colab.

Conclusion:
While I didn't have many expectations when starting this project, the ones that I did have were very consistent with the results. The one big expectation that I had coming in was that the Naive Bayes algorithm would be a sort of baseline for the other algorithm to be compared to and show how it was better.

While I don’t specifically plan to continue doing personal research in the network area of computer science, I do plan to continue on with research regarding machine learning.

This project helped me learn many things. This includes how to use Google Colab and Jupyter Notebook, how to display graph data, how to use machine learning libraries, and how to use machine learning algorithms.

Carter Webb
CPE 400

Works Cited

Numpy. “NumPy.” *Numpy.org*, 2009, numpy.org/.

Pandas. “Python Data Analysis Library.” *Pydata.org*, 2018, pandas.pydata.org/.

Matplotlib. “Matplotlib: Python Plotting — Matplotlib 3.1.1 Documentation.” *Matplotlib.org*, 2012, matplotlib.org/.

Scikit-learn. “Scikit-Learn: Machine Learning in Python.” *Scikit-Learn.org*, 2019, scikit-learn.org/stable/.

TQDM, Casper da. “Tqdm Documentation.” *Tqdm.github.io*, tqdm.github.io/.

Imblearn. “Imbalanced-Learn Documentation — Version 0.8.1.” *Imbalanced-Learn.org*, 2021, imbalanced-learn.org/stable/.

TensorFlow. “TensorFlow.” *TensorFlow*, Google, 2019, www.tensorflow.org/.

Kaggle. “Public API Documentation.” *Www.kaggle.com*, www.kaggle.com/docs/api.

freeCodeCamp.org. “Machine Learning for Everybody – Full Course.” *YouTube*, 26 Sept. 2022, www.youtube.com/watch?v=i_LwzRVP7bg.