Alenna Zweiback
Project #2

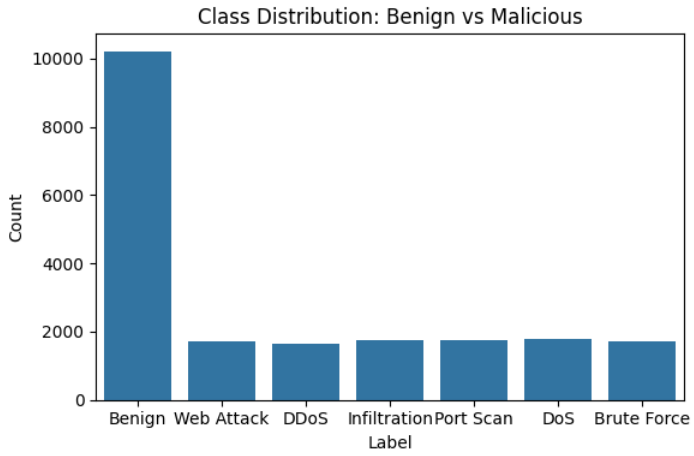
Project Description

The objective of this project was to design, develop, and evaluate a machine learning model that is able to detect network attacks. In this project, we were particularly focused on attacks affecting the availability aspect of cybersecurity, such as Denial-of-Service (DoS) attacks. The goal was to distinguish malicious network traffic from benign traffic using tree-based classification models trained on a labeled dataset of packet data.

Exploring the Data

We were provided with a labeled training data set that had more than 45,00 samples of network packets. Each sample had 1,525 features and was labeled either "benign" or malicious. Each row represented a single network packet and had one label column indicating the type of traffic. During exploration, I used .shape and .info() to verify the structure and datatype composition of the dataset. My Colab showed the training dataset to have of 28,564 samples and 1,526 columns. The dataset contained mostly float-type features, and a class distribution analysis showed a mix of benign and malicious categories. In the malicious category, it contained subcategories for different types of attacks such as "Web Attack," "DDos," "Infiltration," "Port Scan," "Brute Force," and "DoS." I created a class distribution to visualize the imbalance, where I found that "Benign" was significantly higher than all other attack categories.

Label	#
Benign	10215
DoS	1797
Infiltration	1739
Port Scan	1736
Brute Force	1723
Web Attack	1705
DDoS	1644



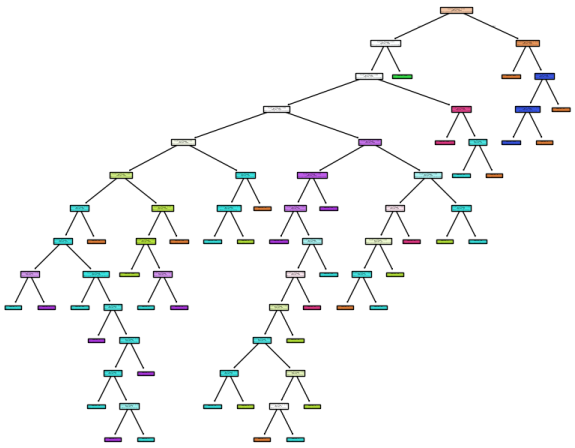
I continued to explore the data and used describe() and found that most features were already normalized between 0 and 1. I also checked for missing values using .isnull().sum() and found a few rows with nulls in the last few bytes, including the label column. These rows were removed using df.dropna(inplace=True) to prevent any disruption during model training. I did not alter the data set any further, as it seemed ready to test a few classification tree models.

Preprocessing & Splitting Techniques

After I explored the data, I extracted the features (X) and labels (y) using indexing from prior knowledge, and selected all but the last column for X and used the last column for y. I split the data into training and testing subsets using an 80/20 split. I chose this ratio because it has been a standard practice of mine in past projects, and I feel like it gives the best balance for evaluation.

Classification Models

In the first model, I implemented a Decision Tree Classifier. In this model, it splits the data based on feature values to make decisions. It tends to be prone to overfitting but it is useful for understanding how the model learns. I initially trained it using ccp_alpha = 0 so no pruning was applied. To understand how pruning affected model performance, I tested several values of ccp_alpha including 0.01, 0.1, 1, and 2. The best accuracy was achieved with ccp_alpha = 0 at 99.24%. As pruning increased, accuracy dropped significantly. It went down to 64.01% at ccp_alpha = 0.1, and to 46.09% at ccp_alpha = 1 and 2. It helped to confirm that minimal or no pruning retained more decision boundaries, leading to better fit for this dataset. These results confirmed that little to no pruning allowed the tree to retain more decision boundaries, which led to a better fit for the training data. The Decision Tree was evaluated using accuracy, a confusion matrix, and a full classification report that included precision, recall, and F1-score for each class.



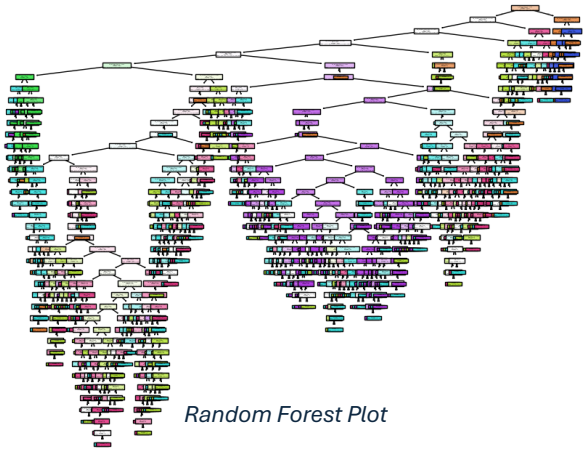
Decision Tree Model

Predicted	Benign	Brute Force	DDoS	DoS	Infiltration	Port Scan	\
Actual							
Benign	2024	0	0	0	0	0	
Brute Force	0	364	0	2	0	0	
DDoS	0	0	316	0	0	0	
DoS	0	3	0	326	0	1	
Infiltration	1	0	0	0	396	0	
Port Scan	0	0	0	2	0	336	
Web Attack	0	0	0	0	0	0	

Predicted	Web Attack
Actual	
Benign	1
Brute Force	0
DDoS	0
DoS	0
Infiltration	0
Port Scan	0
Web Attack	340

Confusion Matrix for Decision Tree

The second model was a Random Forest Classifier. It is an ensemble of Decision Trees and improves accuracy and reduces overfitting by combining multiple trees trained on different subsets of data. I experimented with different values for n_estimators, including 50, 90, 95, 100, 120, 150, and 200, to assess how the number of trees impacted performance. The model achieved its highest accuracy (98.48%) with several of these values, including 90, 100, and 200 trees. The Random Forest performed very well, and it showed more stability and generalization compared to the single Decision Tree. I evaluated this model using accuracy, confusion matrix, and classification report.



Random Forest Plot

Predicted	Benign	Brute Force	DDoS	DoS	Infiltration	Port Scan	\
Actual							
Benign	2021	2	2	0	0	0	
Brute Force	0	351	0	15	0	0	
DDoS	4	0	312	0	0	0	
DoS	0	0	0	330	0	0	
Infiltration	1	0	0	0	396	0	
Port Scan	0	0	0	3	0	335	
Web Attack	0	0	0	7	0	0	

Predicted	Web Attack
Actual	
Benign	0
Brute Force	0
DDoS	0
DoS	0
Infiltration	0
Port Scan	0
Web Attack	333

Confusion Matrix for Random Forest

As a bonus, I also tested a Logistic Regression model. It is a linear model that estimates the probability of class membership. It is useful as a baseline but less effective for complex or non-linear patterns. Although not required, I included it to benchmark how a linear model would perform against the tree-based models. I ran Logistic Regression with max_iter set to 100, 500, 1000, 2000, and 5000. The model converged in all cases, and it produced consistent results with an accuracy of 91.79% across all iterations. While this was lower than the tree-based models, it was baseline and confirmed that the other models were better.

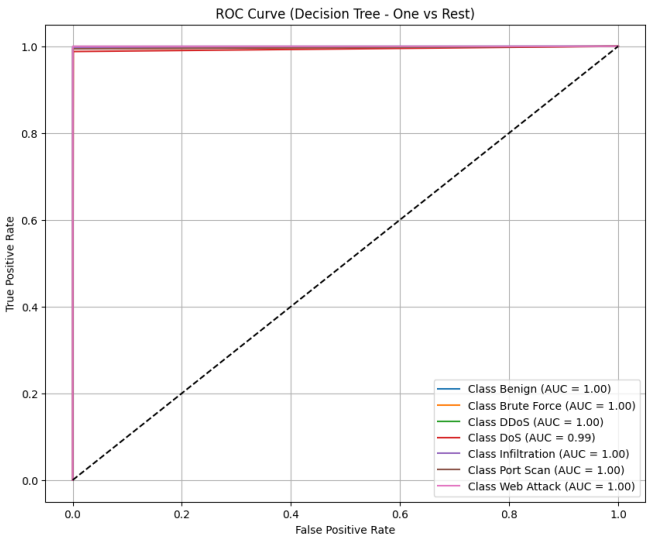
Model Performance

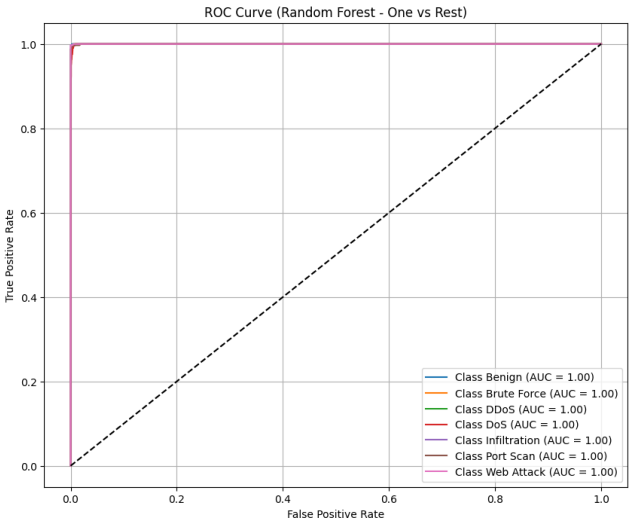
All of the models were evaluated using accuracy, confusion matrix, and a classification report that provided precision, recall, and F1-score for each class. I choose to evaluate these metrics to measure both overall correctness (accuracy) and class-specific performance (precision/recall/F1) which is especially important due to class imbalance I saw earlier. Below is a comparison of the Accuracy and F1 Score across the three models:

Model	**Accuracy**	**F-1**
Decision Tree	0.9975	0.9975
Random Forest	0.9917	0.9917
Logistic	0.9474	0.9478

ROC Curves

To further evaluate model performance, I generated ROC curves for both the Random Forest and Decision Tree classifiers using a one-vs-rest approach. The Random Forest model achieved near-perfect results, with all classes showing an AUC of 1.00 which indicated excellent separability between classes. The Decision Tree also performed strongly but slightly less consistently. While most classes reached an AUC of 1.00, the DoS class had a slightly lower AUC of 0.99. This suggests that although both models performed well, Random Forest was more consistent in distinguishing between all attack types and benign traffic. The graphs are seen below:





Conclusion

After evaluating all three models, I compared their performance side by side. The Decision Tree achieved the highest raw accuracy at 99.75% but showed signs of overfitting especially when I saw how much performance dropped with even slight pruning. The Random Forest showed slightly lower accuracy at 99.17% but was more robust across different parameter settings which suggested better generalization. Logistic Regression performed significantly worse than both tree-based models which confirmed my prior belief that the relationships in the data were too complex for a linear classifier.

Based on this analysis, I selected the Random Forest model (with n_estimators=90) as the best-performing classifier. I used it to generate predictions on the unlabeled test dataset. I also chose Random Forest because it handles high-dimensional data well and it will require minimal tuning to achieve strong results, which makes me confident that the predictions will generalize well. Even though there were some misclassifications in the confusion matrix compared to the Decision Tree, I believe that Random Forest will still be the more reliable model overall due to its stability, stronger generalization to unseen data, and reduced tendency to overfit.