CS 4641 Project Final Report - Cybersecurity Analysis

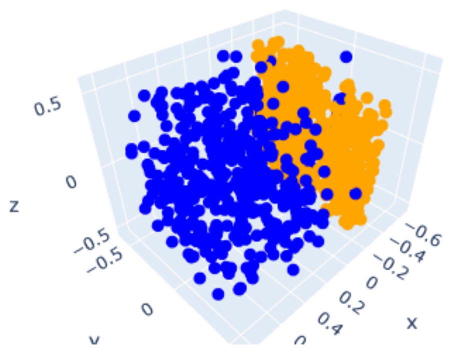
Introduction & Background

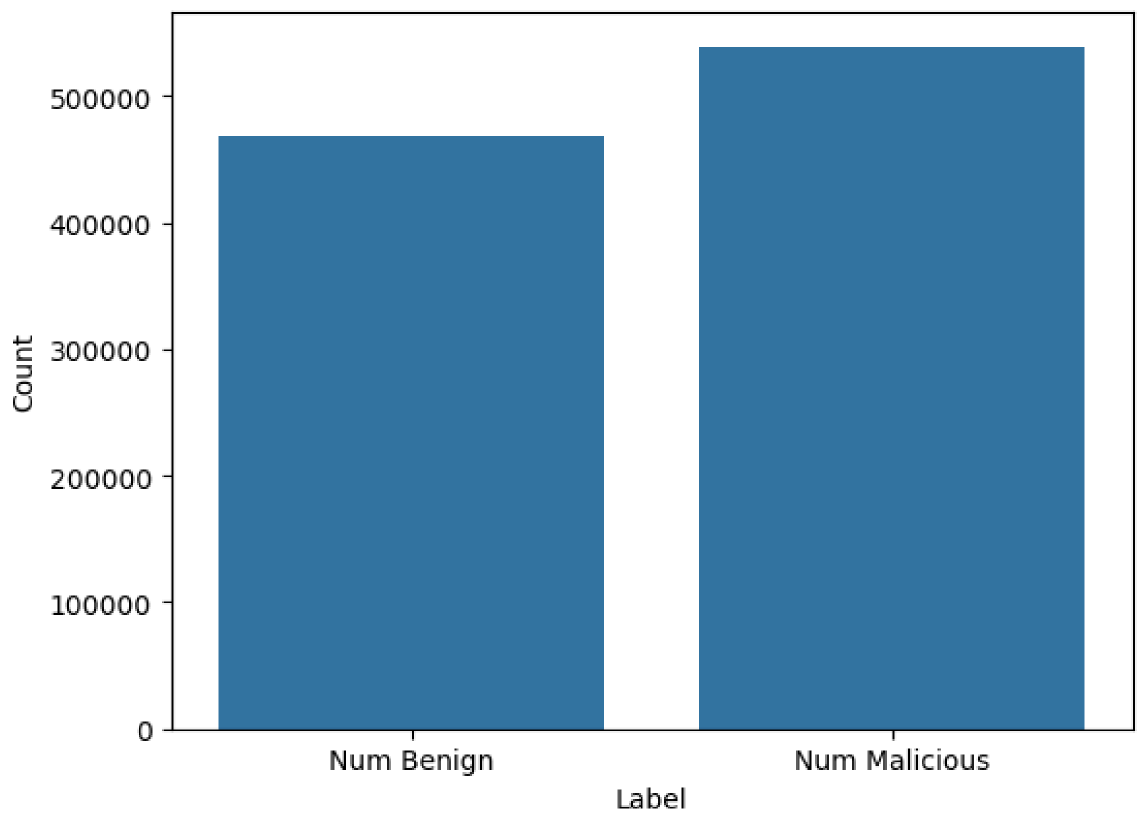
The spread of the internet has given rise to malicious attacks meant to compromise computer systems. This has brought about the field of cybersecurity, concerned with defending electronic systems, networks, and data [1]. However, as cybersecurity systems become more advanced, so do the attacks they prevent, necessitating systems that learn to prevent new attacks themselves [1]. Currently, researchers use machine-learning techniques to identify new threats. Decision trees, convolutional neural networks, and support vector machines are most effective at detecting malware based on log data [2].

Our dataset provides network traffic data labeled as benign or malicious. The features include source and destination IPs and ports; network protocol; associated service; duration; state of connection; whether it's local; number of packets, bytes,and IP bytes sent; number of missed bytes; and whether the connection is malicious. The dataset was obtained from Kaggle: www.kaggle.com/datasets/agungpambudi/network-malware-detection-connection-analysis [3].

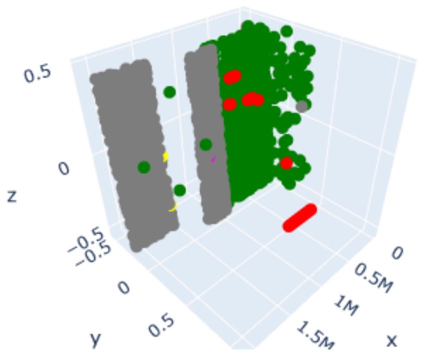
As changed from the midterm, we modified the dataset that we used for training. At first, we used a simple, single csv file as our benign-malicious dataset. We used this dataset to train our initial linear SVM model, but decided it was too simple. As such, we merged the remaining csv files into one composite dataset, which contained many more labels and many more datapoints to learn while maintaining the same features. The merged dataset had many classes, which were simply merged into just benign or malicious, due to the imbalanced frequency of classes. The SVM analysis will showcase two results, one for the initial dataset and one for the binary-labeled merged dataset, while the random forest and neural network models were solely trained on the merged dataset. The datasets look like the following:

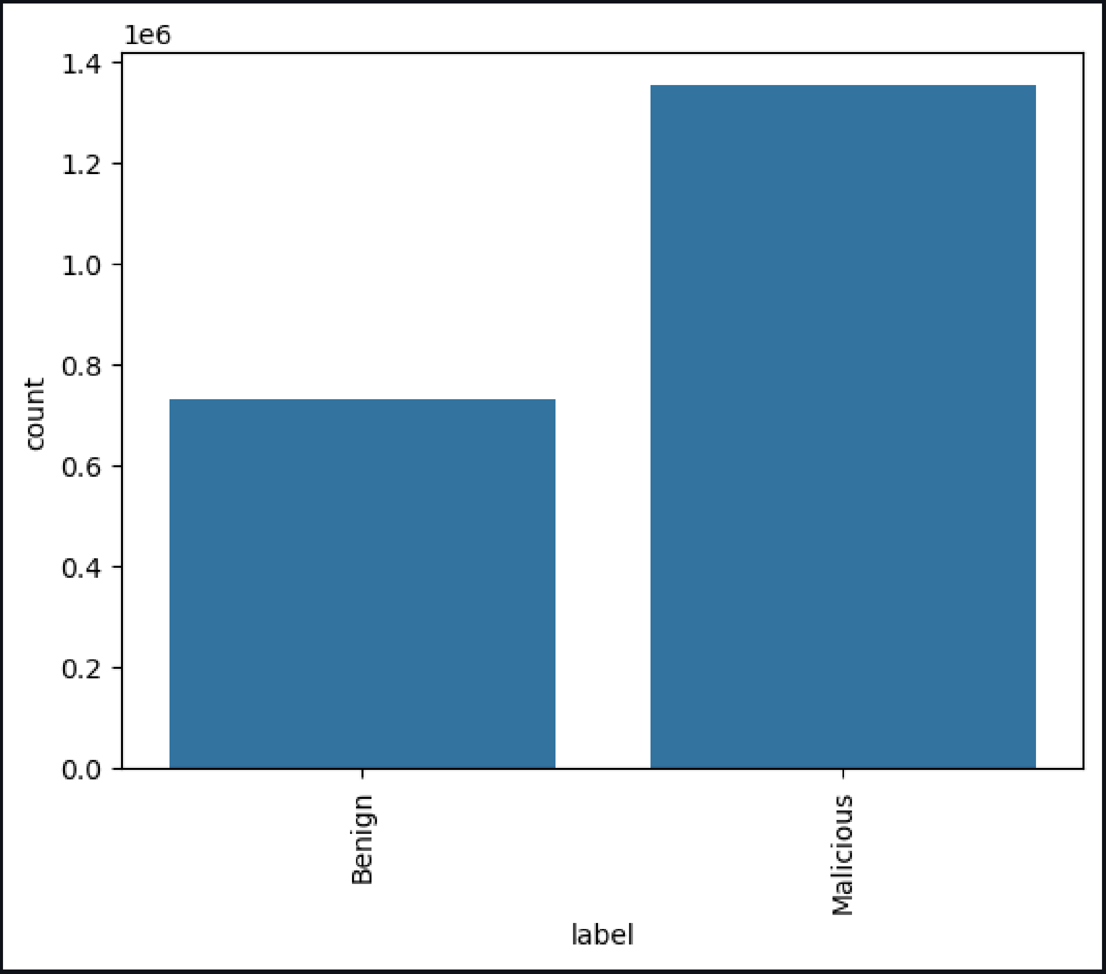
Initial Simple Dataset





Merged/Composite Dataset





Problem Definition

Cybersecurity attacks, specifically network attacks, are becoming increasingly prevalent [4]. A lot of these attacks are through the network, where malware can be remotely delivered, accessed, updated, and executed. The first step in mitigating such attacks is identifying whether or not network activity is benign or malicious. We aim to identify such attacks, so further defense mechanisms (e.g. firewalls, challenges, cleaning infected devices) become more effective and efficient.

Methods

One preprocessing method, often used for network intrusion detection systems (NIDS), is normalizing non-categorical data to within a range from 0 to 1 [5]. This reduces outlier impact and feature weights dominating the learning process. The dataset is then standardized with a mean of 0 and a standard deviation of 1, which makes the data more consistent. These techniques resulted in significant improvements in accuracy for models like K-nearest neighbors and naive Bayes models. We implemented this using `sklearn.preprocessing.MinMaxScaler`.

Another method is lightweight feature extraction, specifically, Correlation-based Feature Selection (CFS). The process selects a subset of features with the highest correlation with the observed variable and the least correlation with each other [6]. Network features are often redundant, so this helps with dimensionality reduction.

The third preprocessing method is label encoding, which converts categorical labels into numbers. Since our dataset has qualitative data, this approach ensures all features are considered. We implemented this using `sklearn.preprocessing.LabelEncoder`.

We also used PCA, which we did not mention in our proposal, to transform our data and perform basic dimensionality reduction. By analyzing the components and the contributing features, we were able to drop a feature or 2 from our data. We use `sklearn.decomposition.PCA`.

We propose three supervised ML algorithms: Support Vector Machine (SVM), Random Forest, and Convolutional Neural Networks (CNN). So far, we have implemented a model based on SVM.

We implemented SVM as our first algorithm. Initially, we chose SVM due to its effectiveness with high-dimensional data, but this benefit did not come into play since we used PCA and our own knowledge of networking to reduce the number of dimensions. We also opted for a linear kernel at first, aiming to establish a straightforward decision boundary. While this worked great for our initial simple dataset, the linear kernel did not work well for the more complicated merged dataset. So, we trained an SVM using the Radial Basis Function (RBF) kernel, which worked relatively well. All implementations were done with `sklearn.svm`.

Secondly, we expect that Random Forest will be a good algorithm to use because it allows us to identify the most important features to identify malware by providing feature importance scores [8]. The Random Forest algorithm performs well with imbalanced datasets, which is often the case with network logs. We will use the `sklearn.ensemble.RandomForestClassifier` library for implementation.

Finally, we expect that ANNs will be a good algorithm to use. We decided to use Artificial Neural Networks instead of our original selection of Convolutional Neural Networks due to the tabular nature of our data. The datasets identified do not have a spatial bias that image data tends to have, so an ANN's simpler structure should properly handle the data's features with increasing efficiency. Since ANNs do not have convolutional layers and instead have layers with fully connected neurons, they can process input features without needing to evaluate spatial relationships [9].

Results and Discussion

We utilized PCA for dimensionality reduction and to help visualize our dataset. For the initial dataset, we found that the data is highly learnable, with clear boundaries between classes for the most part. The dataset used was also relatively balanced, so we did not have to worry about corresponding issues.

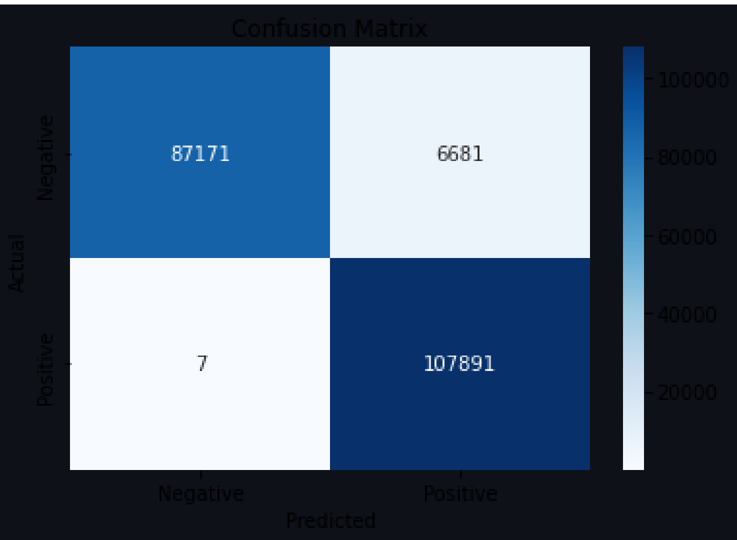
As mentioned previously, we used this initial, simple dataset for our initial SVM implementation, but went ahead and created a composite dataset to further train a non-linear SVM, ANN, and Random Forest models.

Our expected results for detecting malicious network traffic show strong performance across three models.

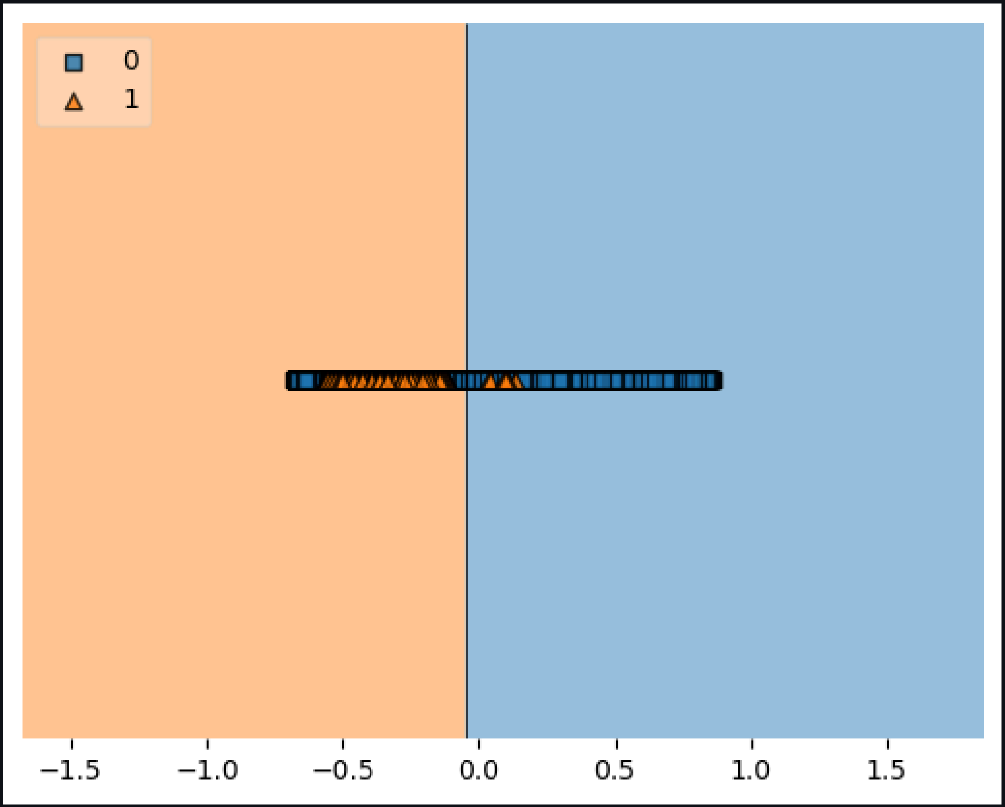
SVM Results & Discussion

SVM was anticipated to achieve 90-95% accuracy, with precision between 0.85-0.92 and recall from 0.88-0.95, excelling at identifying high-dimensional data [7][9]. We increased our expected accuracy from the previous 80-85% projected accuracy due to the learnable nature of the initial dataset. Previous users have been able to achieve near 100% accuracy without needing to finetune their models. For our merged dataset, we expect marginally lower results, with 80-85% accuracy and maintaining the same expected precision and recall.

Initial Simple Dataset Results



Linear SVM Simple Data Confusion Matrix



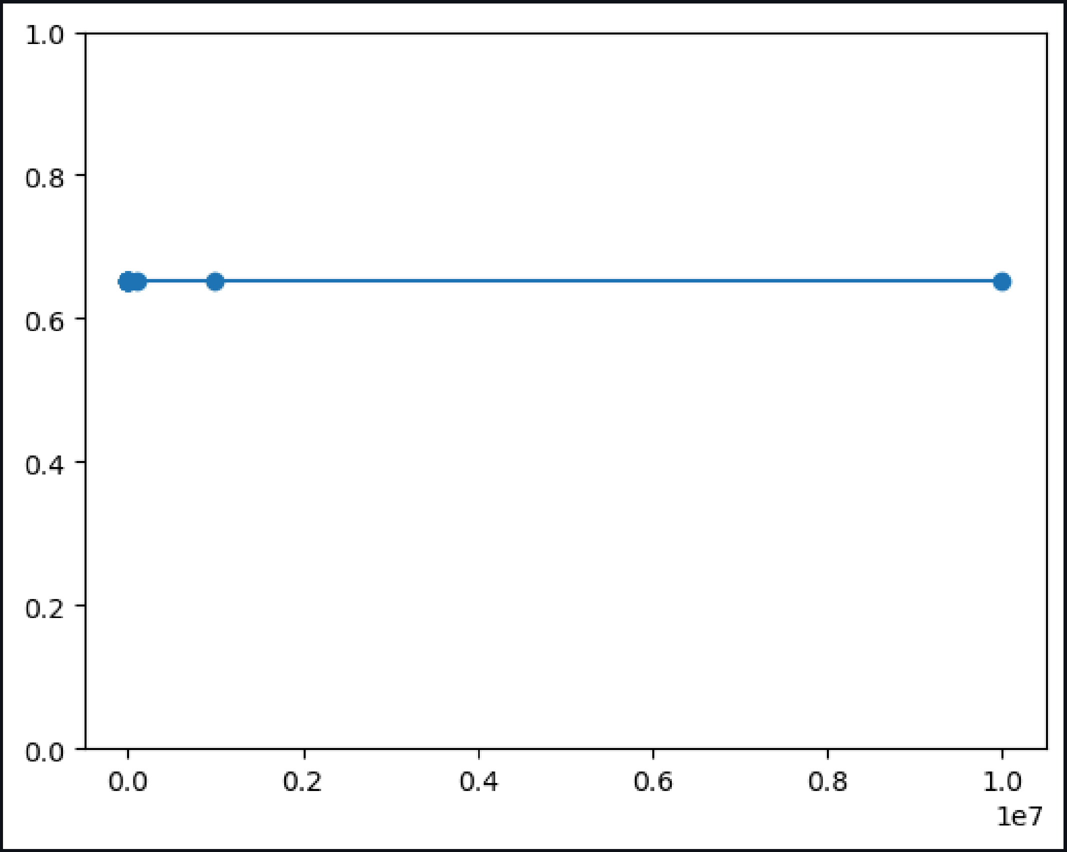
Linear SVM Simple Data Decision Boundary

Accuracy: ~0.967

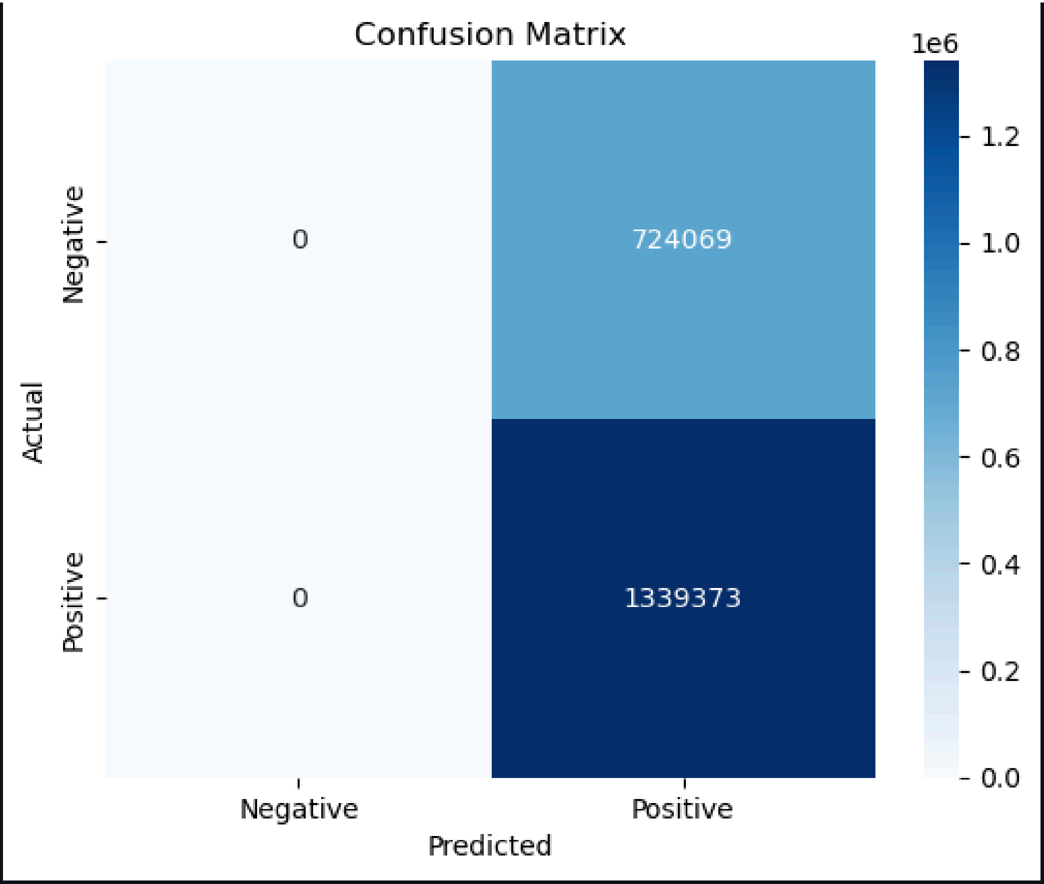
Our SVM model has a fairly high overall accuracy, however it produces a lot of false positives, so predicting a network connection is malicious more often than it actually is. For this case, false positives is definitely a better problem to have than false negatives. For example, it would be fine to block benign connections if that means we also block actual malicious connections (especially since we can further reduce impact on benign connections by first sending challenges before blocking outright). The model did not end up being very inefficient, despite the large dataset, likely because of the pre-processing done to remove features and normalize data. We found that just one component was sufficient to score such an accuracy.

We also had precision ~0.942 and recall ~0.9999, with a false positive rate of ~0.0712. This aligns with our findings of relatively high false positive rates, but overall very good performance.

Merged/Composite Dataset Results



Linear SVM Cross Validation

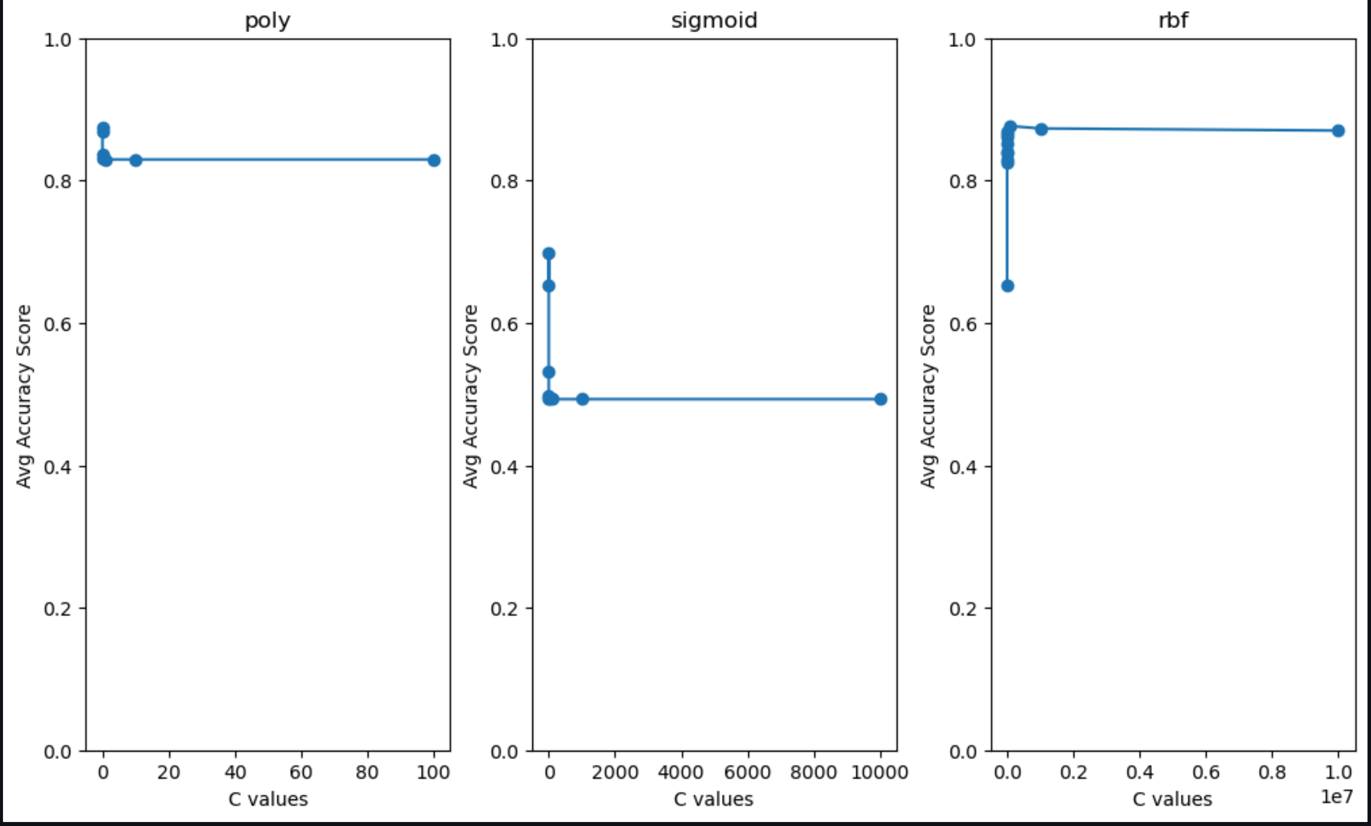


Linear SVM Confusion Matrix

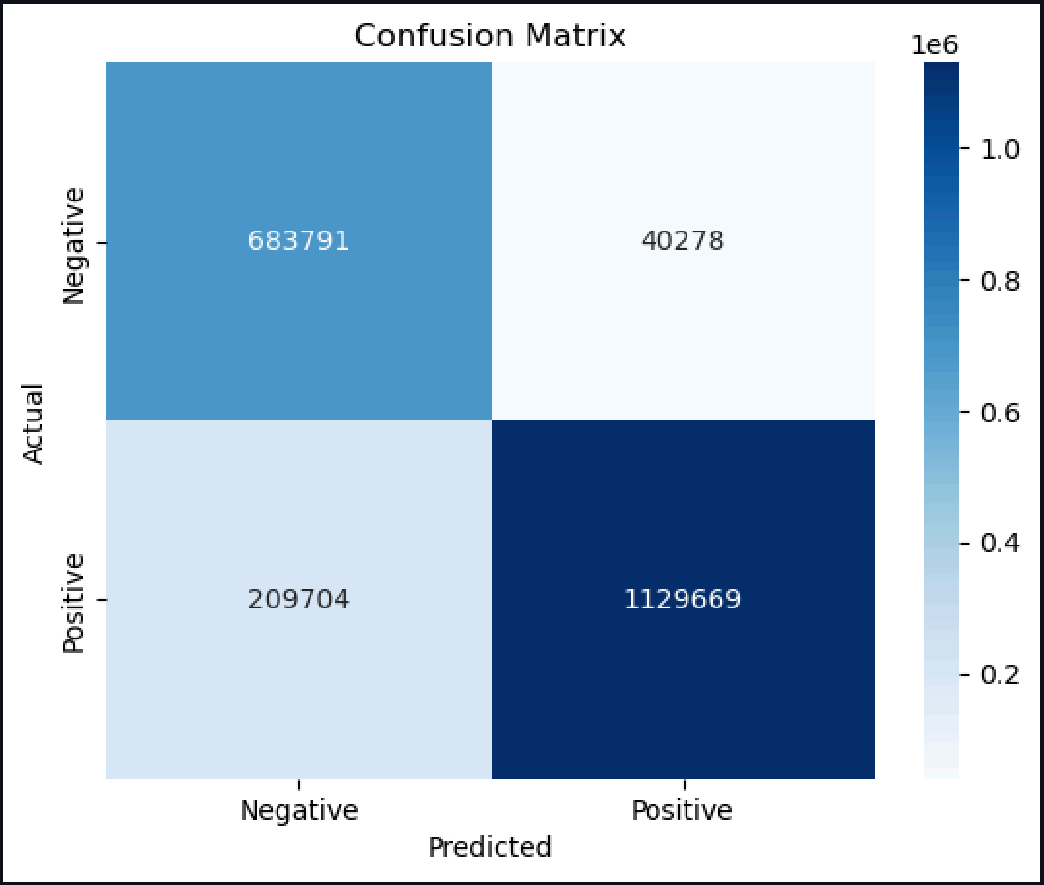
For the generalized dataset, a linear SVM did not perform well, with a 65% accuracy. The accuracy is only greater than 50%, since the dataset is imbalanced, with more malicious datapoints than benign, and the model predicts everything as malicious. This lackluster performance is expected, since the visualization of the dataset shows a nonlinear separation between benign and malicious datapoints.

So, we trained multiple non-linear SVMs with various C values and kernels, with the RBF kernel at C-value 100,000 performing the best in validation. The polynomial kernel was next best with near-zero C values, and the sigmoid kernel performed worse than the linear model.

We achieved an 87.9% accuracy, 96% precision, and 84% recall. With the RBF SVM model, there is a surprisingly low false positive rate of 0.0556 (meaning few benign connections are marked malicious), but relatively high false negative rate of 0.1566(malicious connections are marked benign). The false negative rate is definitely a problem, as we do not want to have malicious connections go through. This is likely due to the limitations of the SVM and the kernel, since dataset visualization shows separate “chunks” of benign and malicious datapoints.



RBF SVM Cross Validation

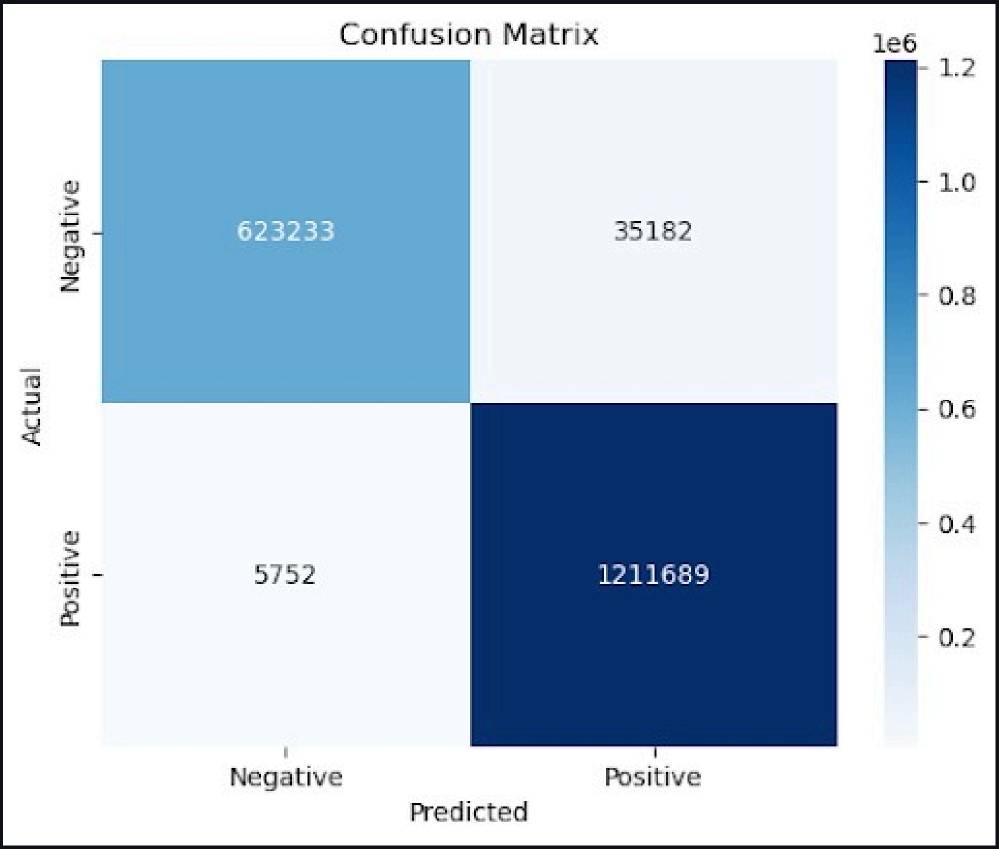


RBF SVM Confusion Matrix

	Metric	Percentage
0	Accuracy	0.879
1	Precision	0.966
2	Recall	0.843
3	False Positive Rate	0.0556
4	False Negative Rate	0.1566

As expected, training SVMs was also not very practical. Since training scales with the number of datapoints squared, it takes hours or days to train with even a few hundred thousand datapoints. As such, it required finding the right number of datapoints to learn good patterns from while also keeping training time reasonable. Also, finding the right kernel takes time, both in understanding the patterns of data and researching the appropriate kernel. Training an ensemble of linear SVMs may be a suitable alternative, both for training and performance. Even though an SVM can be modified to work for non-linearly separable data, it has too many performance and training limitations.

Random Forest Results & Discussion



Random Forest Confusion Matrix

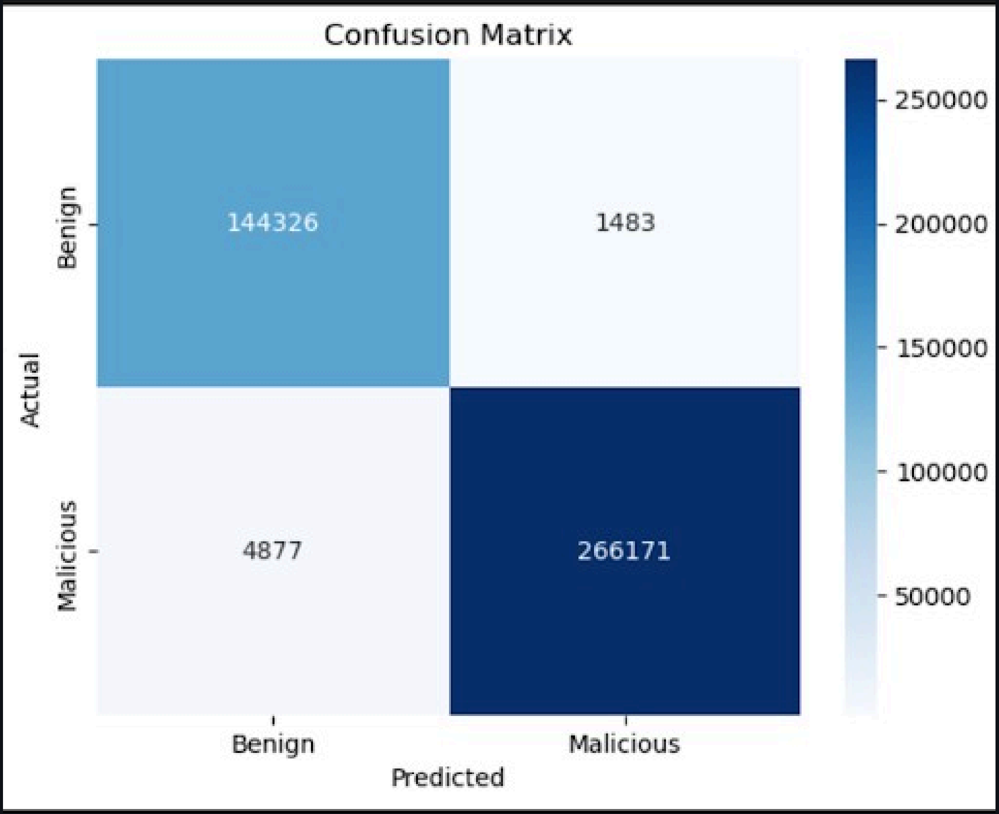
	Metric	Percentage
0	Accuracy	0.978
1	Precision	0.972
2	Recall	0.995
3	False Positive Rate	0.053
4	False Negative Rate	0.005

Random Forest was expected to reach 93-97% accuracy, with precision and recall both ranging from 0.92-0.96, thanks to its ability to handle imbalanced datasets and provide feature importance [5].'

The random forest model performs very well with the generalized dataset, doing better than expected across all metrics. The rate of false positives is also much higher than false negatives, meaning that even when the model predicts incorrectly, the chance of malicious connections going through is very low. Random Forest likely performed much better than SVM due to its ability to handle imbalance datasets.

Hyperparameter tuning also did not seem to come into play as much, for example the number of features per estimator did not change the average accuracy more than a fraction of a percent.

Artificial Neural Network (ANN) Results & Discussion



ANN Confusion Matrix

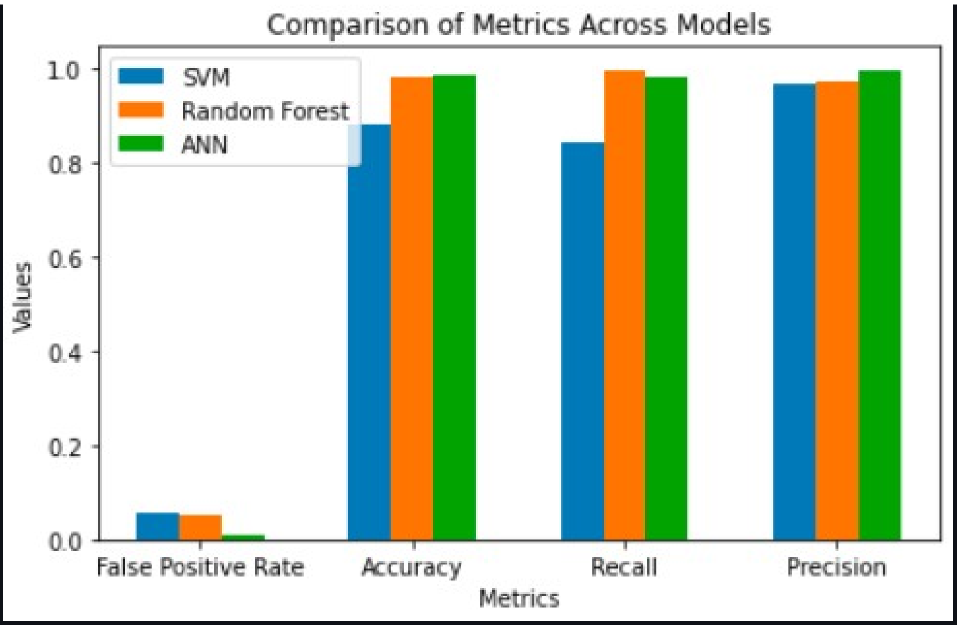
	Metric	Percentage
0	Accuracy	0.986
1	Precision	0.994
2	Recall	0.982
3	False Positive Rate	0.010
4	False Negative Rate	0.018

We expected that the ANN should achieve 94-98% accuracy, with precision around 0.90-0.95 and recall from 0.88-0.94, since it excels at pattern recognition in network traffic which makes it suitable for intrusion detection [10].

The final test accuracy achieved was 98.6%. The neural network performed fairly well with the generalized dataset, and surpassed the desired goal of 94-96% accuracy. It also achieved a precision of 0.994 and recall of 0.982. The false positive rate was 0.01. The model had a fairly low amount of both false positives and false negatives, which could be attributed to the model's simple architecture which could effectively capture the nonlinear patterns without overfitting to the training data. The loss function of binary cross entropy was also changed from the original function of categorical cross entropy, which was well-suited to separate the classes in the feature space.

Several different approaches were taken to try increasing the overall accuracy, which was around 92.6% at first. However most of these attempts resulted in the model plateauing around 30 epochs, at which point the loss began to fluctuate instead of decrease. Firstly, Batch Normalization was added between the layers to normalize the activations of each layer and improve stability during training. However, this caused the model to improve at a much slower rate, and it was only able to achieve a final accuracy of 89% after training for 100 epochs. Drop layers were also added to combat overfitting, however this resulted in the model stabilizing around 84% accuracy, which was reflected in the test accuracy of 86%. There was also an attempt to add varying descending layer width, with 3 layers going from 200 → 64 → 10 nodes. However, this further led the model to stabilize around 40 epochs and remain around 88% accuracy for the remaining epochs. The loss also stopped decreasing early on during training. Some reasons for this behavior could be that the dataset was fairly sensitive to changes in the model architecture, and a simpler, linear structure was better for avoiding overfitting and allowing for more steady, gradual changes to improve the accuracy.

Models Comparison



Model Metrics Comparison

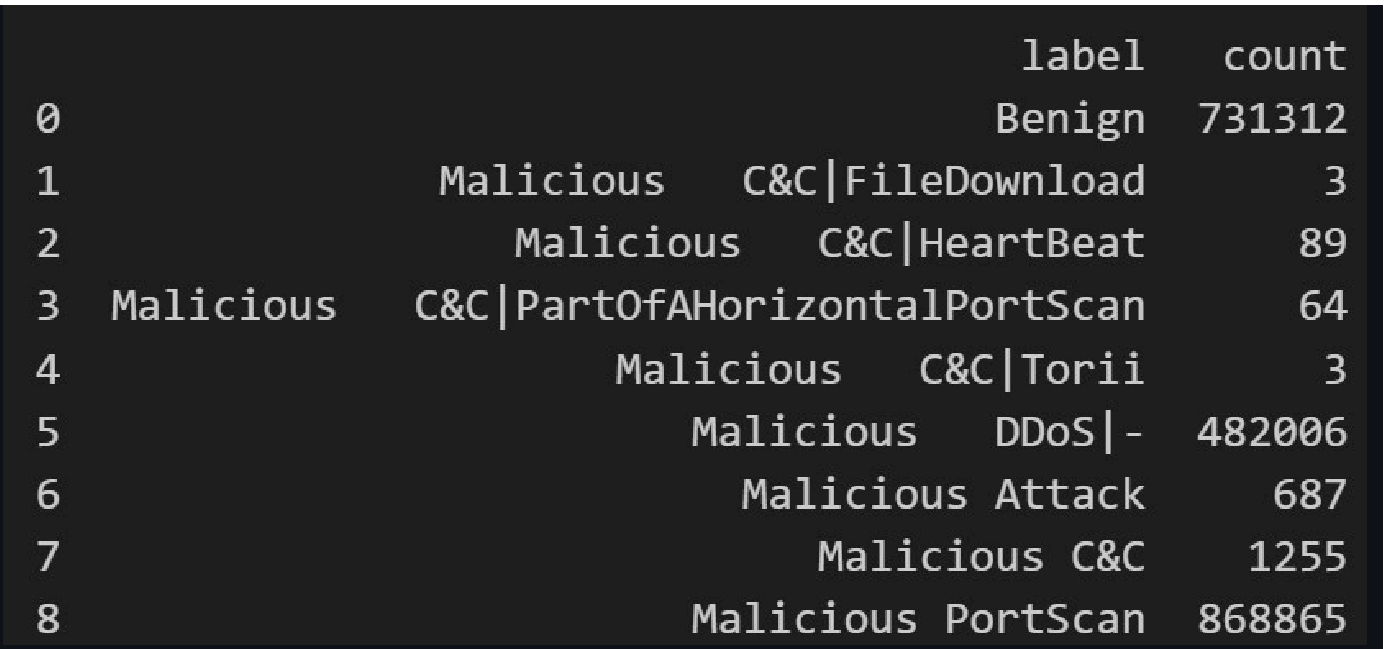
ANN (Model 3) has the lowest FPR (0.01), which is significantly better than both SVM (0.0556) and Random Forest (0.0534). This indicates that ANN is best at minimizing false positives. ANN again has the highest accuracy (0.986), followed by Random Forest (0.978) and SVM (0.879).

ANN is the most reliable for overall correct predictions, and SVM lags behind. Random Forest (0.995) slightly outperforms ANN (0.982) in recall, while SVM (0.843) has the lowest recall.

This suggests Random Forest is slightly better at identifying true positives compared to ANN, while SVM struggles here. All in all, though the Artificial Neural Network implementation performed the best, it's computational expense is a critical factor, and with Support Vector Machines performing the worst, Random Forests might be the way to go considering their extensive use in handling imbalance data.

Next Steps

For our next steps, we could improve the model's ability to categorize the predicted malicious data into specific types, such as malicious file downloads, distributed denial of service attacks, command and control attacks, and other types of security breaches. We have already implemented our goal of using samples of data from numerous different datasets instead of arbitrarily using one dataset, however this approach still had a fairly unequal distribution of data:



```
                                              label   count
0                                            Benign  731312
1                Malicious   C&C|FileDownload          3
2                   Malicious   C&C|HeartBeat         89
3  Malicious   C&C|PartOfAHorizontalPortScan          64
4                       Malicious   C&C|Torii          3
5                          Malicious   DDoS|-     482006
6                            Malicious Attack        687
7                               Malicious C&C       1255
8                          Malicious PortScan     868865
```

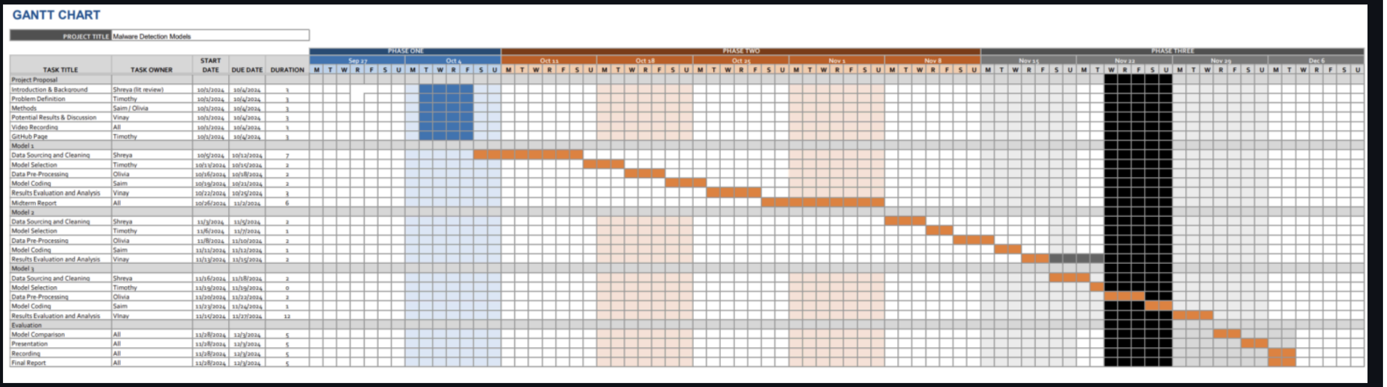
In order to train our model to detect more specific examples of malicious behavior, we need to incorporate more examples of the underrepresented groups of data across a collection of datasets. We should also consider implementing a hierarchical classification structure which first identifies whether an instance of network activity is benign or malicious, and then further classifies it as a specific instance of malicious activity.

Based on the results of our three models, we can also consider improving the environmental ethics and cost-effectiveness of our approach by conserving energy when possible. Neural Networks are more computational expensive than both SVM and Random Forest due to its iterative backpropagation process for recalculating the model weights and needing to pass through multiple layers of neurons in the forward pass. Between SVM and Random Forest, our analysis determined that Random Forest is the more accurate model for network activity classification, so we can primarily focus on optimizing this model by using more feature selection to reduce dimensionality, and implement optimal tree pruning techniques to reduce overfitting and preserve energy. We can also implement an early stopping criteria to further cut down energy consumption and further prevent the model from overfitting to training data.

Incorporating Midterm Feedback and Other Changes

The SVM cross-validation was improved to cover a wider range of C values and kernels for our dataset. We also modified our dataset to be larger and more complex given the data we have, so we retrained the SVM model and used this new dataset for our random forest and ANN implementations.

Gantt Chart



PROJECT TITLE: Malware Detection Models

TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Proposal				
Introduction & Background	Shreya (lit review)	10/1/2024	10/4/2024	3
Problem Definition	Timothy	10/1/2024	10/4/2024	3
Methods	Saim / Olivia	10/1/2024	10/4/2024	3
Potential Results & Discussion	Vinay	10/1/2024	10/4/2024	3
Video Recording	All	10/1/2024	10/4/2024	3
GitHub Page	Timothy	10/1/2024	10/4/2024	3
Model 1				
Data Sourcing and Cleaning	Shreya	10/5/2024	10/12/2024	7
Model Selection	Timothy	10/13/2024	10/15/2024	2
Data Pre-Processing	Olivia	10/16/2024	10/18/2024	2
Model Coding	Saim	10/19/2024	10/21/2024	2
Results Evaluation and Analysis	Vinay	10/22/2024	10/25/2024	3
Midterm Report	All	10/26/2024	11/2/2024	6
Model 2				
Data Sourcing and Cleaning	Shreya	11/3/2024	11/5/2024	2
Model Selection	Timothy	11/6/2024	11/7/2024	1
Data Pre-Processing	Olivia	11/8/2024	11/10/2024	2
Model Coding	Saim	11/11/2024	11/12/2024	1
Results Evaluation and Analysis	Vinay	11/13/2024	11/15/2024	2
Model 3				
Data Sourcing and Cleaning	Shreya	11/16/2024	11/18/2024	2
Model Selection	Timothy	11/19/2024	11/19/2024	0
Data Pre-Processing	Olivia	11/20/2024	11/22/2024	2
Model Coding	Saim	11/23/2024	11/24/2024	1
Results Evaluation and Analysis	Vinay	11/15/2024	11/27/2024	12
Evaluation				
Model Comparison	All	11/28/2024	12/3/2024	5
Presentation	All	11/28/2024	12/3/2024	5
Recording	All	11/28/2024	12/3/2024	5
Final Report	All	11/28/2024	12/3/2024	5

Project Gantt Chart

Contribution Table

Team Member	Contributions
Shreya	Random Forest
Saim	Presentation
Vinay	Model Comparison and Presentation
Timothy	New Dataset and SVM
Olivia	Neural Network and Next Steps

References

[1] J. Martínez Torres, C. Iglesias Comesaña and P. J. García-Nieto, "Review: machine learning techniques applied to cybersecurity," International Journal of Machine Learning and Cybernetics, vol. 10, (10), pp. 2823-2836, 2019. Available: https://www.proquest.com/scholarly-journals/review-machine-learning-techniques-applied/docview/2920238591/se-2. [Accessed Oct. 03, 2024]

[2] M.S. Akhtar and T. Feng, “Malware Analysis and Detection Using Machine Learning Algorithms,” Symmetry, vol. 14, (11), 2304, 2022. Available: https://www.mdpi.com/2073-8994/14/11/2304. [Accessed Oct. 03, 2024]

[3] S. Garcia, A. Parmisano, & M. Jose Erquiaga. (2023). "Malware Detection in Network Traffic Data" [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/7285844

[4] M. Abdullahi et al., “Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review,” Electronics, vol. 11, no. 2, p. 198, Jan. 2022, doi: https://doi.org/10.3390/electronics11020198.

[5] Larriva-Novo, X., Villagrá, V. A., Vega-Barbas, M., Rivera, D., & Sanz Rodrigo, M. (2021). An IOT-focused intrusion detection system approach based on preprocessing characterization for cybersecurity datasets. Sensors, 21(2), 656. Available: https://doi.org/10.3390/s21020656

[6] Soe, Y. N., Feng, Y., Santosa, P. I., Hartanto, R., & Sakurai, K. (2020, January 11). Towards a lightweight detection system for cyber attacks in the IOT environment using corresponding features. MDPI. https://www.mdpi.com/2079-9292/9/1/144

[7] Karaman, Yunus, et al. “A Comparative Analysis of SVM, LSTM and CNN-RNN Models for the BBC News Classification.” SpringerLink, Springer International Publishing, 1 Jan. 1970. Available: https://link.springer.com/chapter/10.1007/978-3-031-26852-6_44. [Accessed Oct. 04, 2024]

[8] Manish Choubisa, R. Doshi, N. Khatri, and Kamal Kant Hiran, “A Simple and Robust Approach of Random Forest for Intrusion Detection System in Cyber Security,” May 2022, doi: https://doi.org/10.1109/icibt52874.2022.9807766.

[9] Vakili, Meysam, et al. “Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IOT Data Classification.” arXiv.Org, 27 Jan. 2020. Available: https://arxiv.org/abs/2001.09636. [Accessed Oct. 04, 2024]

[10] Makandar, A., & Patrot, A. (2015, December 1). Malware analysis and classification using Artificial Neural Network. IEEE Xplore. https://ieeexplore.ieee.org/Xplore/home.jsp