Securing the Internet of Things: Comparing Classification Methods for Adversarial Network Activity Detection

By: Brendan Kenny, Kimberley Maldonado, Emily Su

***Abstract*—Network connectivity is developing at a startling speed as technology advances. As society increasingly depends on an interconnected network of devices, known as the Internet of Things, or IoT, cybersecurity threats become a pressing concern. In this project, we attempt a number of classification model techniques, focusing on ensemble methods and SVM (support vector machine), to assess their effectiveness at identifying both harmful and normal network data behaviors using real-time IoT infrastructure data. The performance of each model included accuracy as well as the macro-averages of precision, recall and F1-score. The results demonstrate that overall the Decision Tree based models were suitable for classifying the network data. The ensemble methods, more specifically the Random Forest and AdaBoost models performed the best.**

Introduction

As technology advances, the proliferation of Internet of Things (IoT) devices and their interconnectedness continues to grow. This increasing network of interconnected devices brings about significant challenges, particularly in terms of security and privacy, which are facing more frequent and sophisticated attacks.

One of the best ways to combat cyber-attacks is by detecting them early through identification of abnormal or atypical traffic through a given network. A method to identify these different types of traffic is through the use of classification, which is commonly used on incoming traffic to determine if that traffic pattern is either atypical or normal based on it's characteristics.

The *RT-IoT2022* dataset [SN24] includes real-time IoT infrastructure data from a diverse range of IoT devices and sophisticated network attack methodologies. In this project we attempt a number of different classification model techniques, specifically ensemble methods, including Decision Tree, Random Forest, KNN, boosting methods such as XGBoost and Adaboost, and Support Vector Machine (SVM), to assess their effectiveness at identifying both harmful and normal network data behaviors.

Related Work

Due to the growing concern of privacy and data loss and increasing importance of cyber-security there is an abundance of research being completed to develop ensemble classifiers for the detection of network attacks. XGBoost and AdaBoost are both tree based ensemble models, specifically boosting models, which combine multiple weak learners sequentially into a stronger one.

In Chauhan and Atulkar [CA21] the effectiveness of decision tree based ensemble methods including Random Forest, LGBM, Extra Tree, Gradient Boost, and XGBoost was tested in identifying harmful network behavior. They evaluated each method by calculating accuracy, F1 score, Recall and Precision. Also, the detection time was used to determine the overall effectiveness of the proposed IDS, intrusion detection system, during a real-time event. LGBM was found to outperform the other ensemble methods when comparing both the performance measuring metrics and the detection time.

The work by Danso et al. [Dan+22] proposed testing all ensemble methods, including boosting, stacking and voting. The following models were combined as an ensemble: Naïve Bayes (NB), Support Vector Classification (SVC), and k-Nearest Neighbors (kNN). Testing each of the base learners individually resulted in accuracy values of 98.6% for NB, 88.77% for SVC, and 96.19% for kNN. By comparison the ensemble methods combining all three models resulted in 99.73% for boosting (gradient boosting), 99.87% for stacking, and 99.64% for voting. The Ensemble methods resulted in better accuracy when compared to the individual base learners. P. K. Danso et al. propose the use of stacking, as it resulted in the greatest overall performance.

Moustafa et al. [MTC19] propose the use of an Adaboost ensemble combining decision tree, Naive Bayes (NB), and artificial neural network as a means of providing a better detection rate and reduced false positive rate when compared to other classification techniques. When comparing the base learners to their combination as an ensemble the accuracy, detection rate, and false positive rate metrics were all improved in the ensemble when tested on two different datasets, UNSW-NB15 and NIMS dataset.

Solution

Description of Dataset

Taken from the UC Irvine Machine Learning Repository, *RT-IoT2022* encompasses both normal and adversarial network behaviors, providing a general representation of real-world scenarios, incorporating data from IoT devices. The normal networks are MQTT, Think Speak, and Wipro_bulb_Dataset, whereas the rest are considered malicious attacks, which can be identified in Figure 1. The bidirectional attributes of network traffic are captured in the dataset. Instances of the data sets include a total of 83 features such as Duration, Number

of packets, Number of bytes, Length of packets, etc. are also calculated separately in the forward and backward direction.

This dataset does not contain any missing values, however, an irrelevant column named 'Unnamed: 0' was removed. Of sizeable concern was that the number of features included in the data was very large (123117 rows × 83 columns). As such, Principal Component Analysis (PCA) was employed to mitigate the computational complexity of the classification task at hand. *StandardScaler* was used to scale all the features to make them within the same range of each other. Another future potential course of action is to group the dataset to increase the running speed of the machine learning algorithms that will be discussed. To validate our models, we used *train_test_split* to use 80% as our training sample and 20% as our testing sample.

A challenge presented by the dataset was the overall imbalance in the number of instances for each class. The *DOS_SYN_HPING* class has 94659 instances of a total 123117 instances, representing the largest class in the dataset.

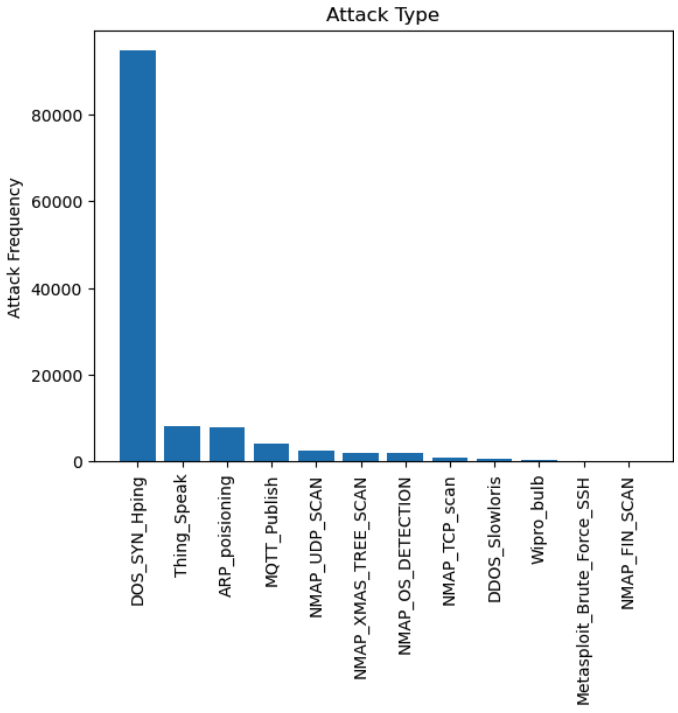


Fig. 1. Bar Graph shows frequency of each Attack Type

The adversarial network attack type frequency can change through time depending on which is most successful. This means that developing a classifier that it not simply best at detecting the most frequency attack type, but the best at identifying each unique attack type. For this reason, and due to the class imbalance within the data, accuracy is not the best metric to evaluate each classifier. A classification report is included for each model which lists the precision, recall and F1-score for each class. Furthermore, by comparing the Macro Average, takes the average of the precision, recall and F1-score of each class the effectiveness of each model can be better compared.

Machine Learning Algorithms

As discussed earlier, we will use the following classification models: decision trees, ensemble method, and SVM to test the malicious and benign cyber-security attacks.

A Decision Tree is a supervised learning algorithm that builds a hierarchical tree structure, starting with the roots, and then branching off multiple times. Finally, a decision tree will result in a root, determining the class of the data instance.

Ensemble methods are the approach to machine learning that involves combining multiple model predictions into one final prediction. Boosting, one of many ensemble methods, involves using multiple weak learners, such as a decision tree, in sequence where each slightly improves the overall accuracy over the previous. XGBoost and AdaBoost are both tree-based ensemble models which are frequently used for classification. AdaBoost stands for adaptive boosting and involves improving each tree from the previous by adjusting the weights of misclassified instances. By contrast, XGBoost or Extreme Gradient Boost is a popular gradient-boosted decision tree machine learning library that adjusts the weights based on reducing a loss function through gradient descent.

Since IoT data can exhibit complex, non-linear relationships between features, tree-based methods such as Decision Trees and Random Forests were of particular interest since they do not assume linearity between feature and target variable. They often provide clear interpretability and can demonstrate robustness in the face of outliers by splitting them into leaf nodes rather than trying to fit them.

Additionally, K-Nearest Neighbors (KNN) is a powerful algorithm that makes predictions about the classification of a data point based on the categories of its nearest neighbors. Unlike decision trees, it does not explicitly learn a model but instead memorizes the training dataset. The value of K represents the number of nearest neighbors to be considered and is a hyperparameter used to compute the distance between data points according to some distance metric such as Euclidean, Manhattan or Minkowski distance.

SVM is a classification technique that is useful when dealing with high-dimensional data. SVM attempts to find the decision boundary, an optimal hyperplane, that separates the data of a specific class from the rest. SVM is applicable for the given dataset because we can identify which features are more effective for normal vs vicious cyber-security attacks. Moreover, we can see whether normal or attack patterns are more populated.

Implementation Details

Our analysis was conducted as follows:

Step 1: The *RT-IoT2022* dataset was loaded directly into Jupityr Notebooks via the UCI Repository, as well as all necessary packages.

Step 2: When processing the data for the Decision Tree, Random Forest, KNN and SVM methods, PCA was used to

perform dimensionality reduction, reducing the computational load on our models while still capturing the maximum variance within the data. Out of the original 83 columns from the dataset, PCA with a desired variance retention of 0.95 kept 32 principal components and produced an Explained Variance Ratio of 95.5%.

Step 3: The aforementioned ML algorithms were implemented with appropriate hyperparameter adjustments to carry out the desired classification task of each benign and attack pattern.

Step 4: An encompassing report containing the precision, recall and F1-score for each class was used to analyze the performance of each model. The macro-average values of the for the precision, recall, and F1-score will be stated. Finally, the overall accuracy of a model will be provided.

Step 5: Confusion matrices were obtained for further evaluation of performance, albeit, more specifically, for the purpose of identifying the classes that were most often mis-classified by each of the respective models.

Results

Decision Tree

Upon performing PCA dimensionality reduction, the data was split into 80% training and 20% test sets and fitted to the model. The resulting Decision Tree accuracy was 99.6%. The following is a table displaying the attack type-by-attack type precision, recall, and F1-score. The results were validated using a confusion matrix, providing a detailed view of the model's performance across various attack types.

Decision Tree Accuracy: 0.9960201429499675

	precision	recall	f1-score	support
ARP_poisioning	0.97	0.98	0.97	1578
DDOS_Slowloris	0.95	1.00	0.98	100
DOS_SYN_Hping	1.00	1.00	1.00	18897
MQTT_Publish	0.99	1.00	1.00	871
Metasploit_Brute_Force_SSH	0.71	0.83	0.77	6
NMAP_FIN_SCAN	1.00	0.67	0.80	3
NMAP_OS_DETECTION	1.00	1.00	1.00	393
NMAP_TCP_scan	1.00	1.00	1.00	220
NMAP_UDP_SCAN	0.99	0.98	0.99	489
NMAP_XMAS_TREE_SCAN	1.00	0.99	1.00	384
Thing_Speak	0.98	0.98	0.98	1625
Wipro_bulb	0.87	0.81	0.84	58
accuracy			1.00	24624
macro avg	0.96	0.94	0.94	24624
weighted avg	1.00	1.00	1.00	24624

Fig. 2. Decision Tree Classification Report

Random Forest

The PCA-transformed data was split similarly for the Random Forest algorithm as it was in the Decision Tree model, with an 80% and 20% split between training and test sets respectively. The RF model produced an accuracy of 99.7%, a slightly better result than the single Decision Tree. Below are the precision, recall and F1-scores for each attack type. The results are consistent with previous implementations of this algorithm in applications regarding Internet of Things and cybersecurity, notably in Sharafaldan et al. [SLG18].

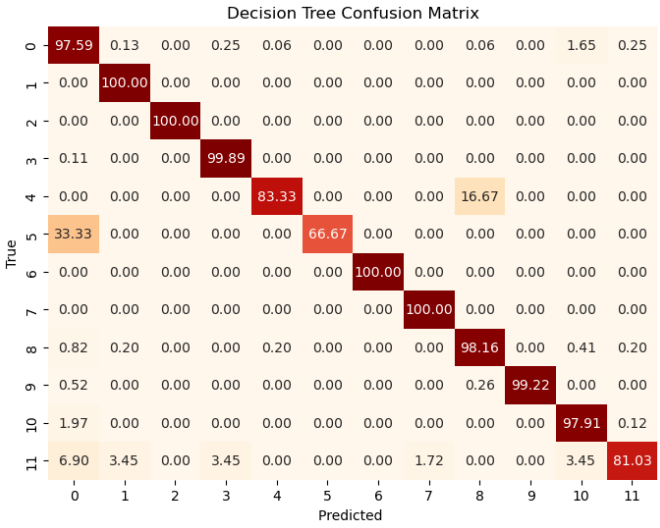


Fig. 3. Decision Tree Confusion Matrix

Accuracy: 0.9971572449642625

	precision	recall	f1-score	support
ARP_poisioning	0.97	0.99	0.98	1578
DDOS_Slowloris	0.99	1.00	1.00	100
DOS_SYN_Hping	1.00	1.00	1.00	18897
MQTT_Publish	1.00	1.00	1.00	871
Metasploit_Brute_Force_SSH	0.83	0.83	0.83	6
NMAP_FIN_SCAN	1.00	0.67	0.80	3
NMAP_OS_DETECTION	1.00	1.00	1.00	393
NMAP_TCP_scan	1.00	1.00	1.00	220
NMAP_UDP_SCAN	1.00	0.98	0.99	489
NMAP_XMAS_TREE_SCAN	1.00	0.99	1.00	384
Thing_Speak	0.99	0.98	0.98	1625
Wipro_bulb	1.00	0.84	0.92	58
accuracy			1.00	24624
macro avg	0.98	0.94	0.96	24624
weighted avg	1.00	1.00	1.00	24624

Fig. 4. Random Forest Classification Report

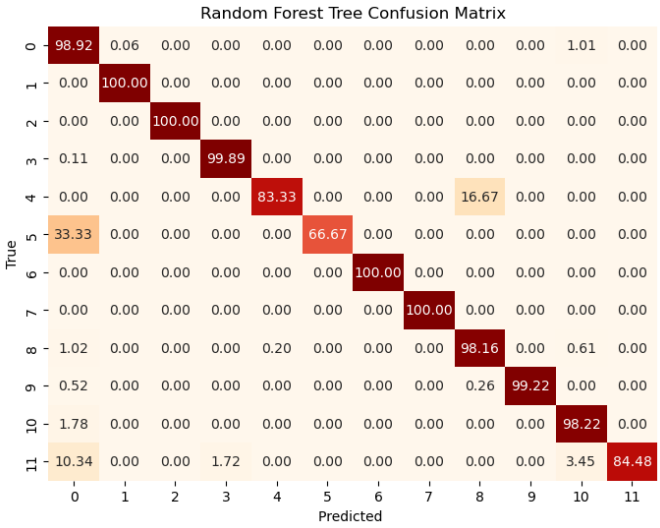


Fig. 5. Random Forest Confusion Matrix

KNN

In our analysis, the hyperparameter K for the KNN model was set to the default value of 5, per scikit-learn's implementation. This model, applied to PCA-transformed data, achieved an overall accuracy of 99.5%, which is comparable to that of the Decision-Tree-based methods. Notably, no explicit hyperparameter tuning, such as K-fold cross-validation, was conducted to determine if a different k value might improve the performance. Furthermore, the class distribution is a vital consideration, as KNN's performance is influenced by the density and distribution in the feature space, which is altered by the use of PCA, potentially affecting the distances between points that KNN relies on. The computational efficiency and scalability of the KNN model, applied to PCA-transformed data, were not explored in this study. Future investigations could benefit from evaluating these aspects to ensure robust and generalizable model predictions.

Accuracy: 0.9949236517218973

	precision	recall	f1-score	support
ARP_poisioning	0.96	0.97	0.97	1578
DDOS_Slowloris	0.98	0.96	0.97	100
DOS_SYN_Hping	1.00	1.00	1.00	18897
MQTT_Publish	0.99	1.00	1.00	871
Metasploit_Brute_Force_SSH	1.00	0.67	0.80	6
NMAP_FIN_SCAN	1.00	0.67	0.80	3
NMAP_OS_DETECTION	1.00	1.00	1.00	393
NMAP_TCP_scan	1.00	1.00	1.00	220
NMAP_UDP_SCAN	0.99	0.98	0.98	489
NMAP_XMAS_TREE_SCAN	1.00	0.99	1.00	384
Thing_Speak	0.97	0.98	0.97	1625
Wipro_bulb	1.00	0.76	0.86	58
accuracy			0.99	24624
macro avg	0.99	0.91	0.95	24624
weighted avg	0.99	0.99	0.99	24624

Fig. 6. KNN Classification Report

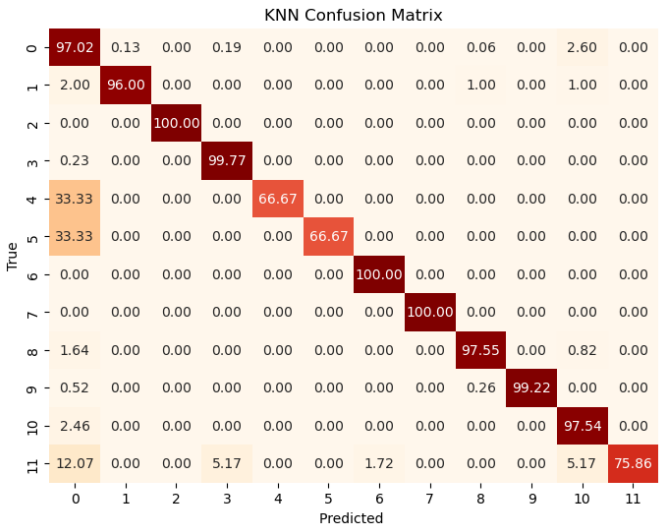


Fig. 7. KNN Confusion Matrix

SVM

Applying a very simple SVM model, using a linear kernel with the *sklearn.svm.svc* implementation, and using all 83 features, we managed to get an accuracy of 75% with the following Precision, Recall, and F1 score of each of the targets.

Attack Type	Precision	Recall	F1
ARP poisioning	0.32	0.21	0.25
DDOS Slowloris	0.00	0.00	0.00
DOS SYN Hping	1.00	0.88	0.94
MQTT Publish	0.80	0.95	0.87
Metasploit Brute Force SSH	0.00	0.00	0.00
NMAP FIN SCAN	0.00	0.00	0.00
NMAP OS DETECTION	0.54	0.89	0.67
NMAP TCP scan	0.08	0.51	0.14
NMAP UDP SCAN	0.01	0.06	0.02
NMAP XMAS TREE SCAN	0.00	0.00	0.00
Thing Speak	0.00	0.00	0.00
Wipro bulb	0.03	0.60	0.05

Using PCA, we have significantly improved the accuracy to 99.2%, achieved by setting kernel to 'poly' and have improved the precision, recall, and F1 score and their macro averages: 90%, 82, 85%.

We also tested various different kernels like poly, rbf, and sigmoid while keeping the other parameters set to default as defined on the Sklearn site. Likewise we noticed the same thing when we tried doing various different test_train splits, specifically the 10/90 and 30/70 split. As such we only reported the highest accuracies as shown in the previous 2 tables, which were produced with a 20/80 split.

To try and further improve the accuracy of the model, we also tested out the other gamma option, *auto*. With this, the accuracy of the *Poly* kernel managed to increase to 99.4%. With this increase we also saw an increase in the macro averages for precision, recall, and F1 score: 95%, 90%, and 92%.

	precision	recall	f1-score	support
ARP_poisioning	0.96	0.98	0.97	2306
DDOS_Slowloris	0.99	0.76	0.86	154
DOS_SYN_Hping	1.00	1.00	1.00	28409
MQTT_Publish	1.00	1.00	1.00	1273
Metasploit_Brute_Force_SSH	0.86	0.55	0.67	11
NMAP_FIN_SCAN	0.60	0.86	0.71	7
NMAP_OS_DETECTION	1.00	1.00	1.00	622
NMAP_TCP_scan	0.99	0.99	0.99	319
NMAP_UDP_SCAN	0.95	0.98	0.97	750
NMAP_XMAS_TREE_SCAN	1.00	0.99	1.00	582
Thing_Speak	0.98	0.98	0.98	2424
Wipro_bulb	0.94	0.86	0.90	79
accuracy			0.99	36936
macro avg	0.94	0.91	0.92	36936
weighted avg	0.99	0.99	0.99	36936

Fig. 8. SVM Classification Report - K = 'Poly', gamma = 'option'

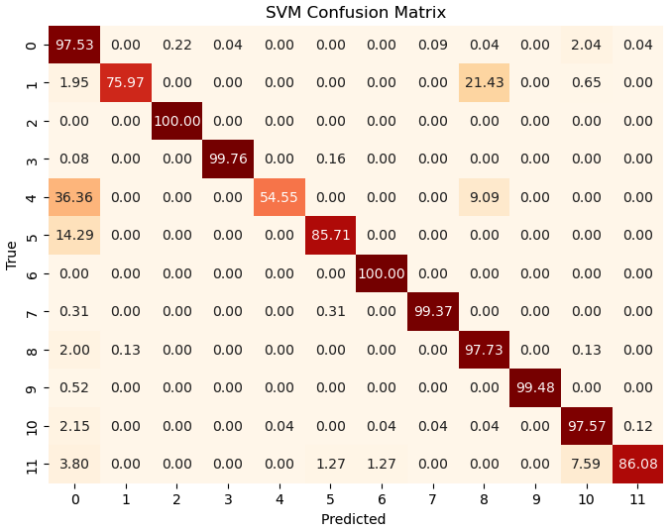


Fig. 9. SVM Confusion Matrix - K = 'Poly', gamma = 'option'

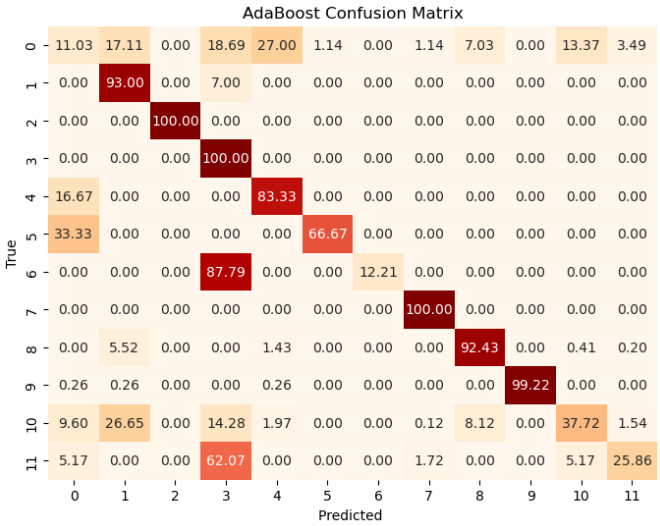


Fig. 11. AdaBoost with PCA Confusion Matrix

Boosting Methods

While PCA was used to pre-process the data for the Decision Tree, Random Forest, and SVM models it was not used for the Boosting ensemble methods. This is because applying PCA to the data prior to training it on boosting methods resulted in lower overall accuracy, as evidenced in Fig. 10. The data was treated with a 80/20 train-test split. Hyper-parmater tuning was performed to identify the optimal model for both XGBoost and AdaBoost. This was done using grid search and randomized search with a 3 fold cross validation.

Accuracy: 0.8841374269005848

	precision	recall	f1-score	support
ARP_poisioning	0.52	0.11	0.18	1578
DDOS_Slowloris	0.11	0.93	0.20	100
DOS_SYN_Hping	1.00	1.00	1.00	18897
MQTT_Publish	0.49	1.00	0.66	871
Metasploit_Brute_Force_SSH	0.01	0.83	0.02	6
NMAP_FIN_SCAN	0.10	0.67	0.17	3
NMAP_OS_DETECTION	1.00	0.12	0.22	393
NMAP_TCP_scan	0.91	1.00	0.95	220
NMAP_UDP_SCAN	0.65	0.92	0.76	489
NMAP_XMAS_TREE_SCAN	1.00	0.99	1.00	384
Thing_Speak	0.74	0.38	0.50	1625
Wipro_bulb	0.16	0.26	0.19	58
accuracy			0.88	24624
macro avg	0.56	0.68	0.49	24624
weighted avg	0.92	0.88	0.88	24624

Fig. 10. AdaBoost with PCA Classification Report

AdaBoost

Applying an AdaBoost model with 100 base estimators and a learning rate of 2.0, with the SAMME algorithm resulted in an accuracy of 84.13%. The AdaBoost function built into scikit-learn defaults the base classifier to a default tree with a max depth of 1. Scikit-learns grid search function was used to perform hyper-parameter optimization on the base estimators, learning rate and max depth hyper-parameters. The grid search was performed with 3 fold cross validation. The scoring was method selected was balanced accuracy, which selects the

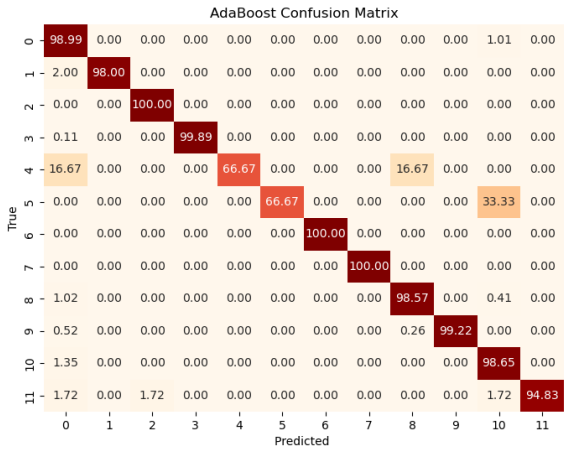


Fig. 12. AdaBoost No PCA Confusion Matrix

hyper-parameters based on taking the average of the recall value for each class. This method is suited for imbalanced data sets.

AdaBoost Grid Search					
hyperparameter	values				
n estimators	50	100	200		
learning rate	0.1	0.5	1.0	2.0	
max depth	1.0	2.0	3.0	4.0	5.0

The optimal model was found to have a learning rate of 0.1 with 100 base estimators and a max depth of 5. This best model resulted in an accuracy of 99.76%, an improvement from the 84.13% accuracy from the original AdaBoost model. The macro average values for Precision, Recall and F1-Score were 99.68%, 93.46%, 96.00% respectively.

AdaBoost Classification Report			
Attack Type	Precision	Recall	F1
ARP poisioning	0.98	0.99	0.98
DDOS Slowloris	1.00	0.98	0.99
DOS SYN Hping	1.00	1.00	1.00
MQTT Publish	1.00	1.00	1.00
Metasploit Brute Force SSH	1.00	0.67	0.80
NMAP FIN SCAN	1.00	0.67	0.80
NMAP OS DETECTION	1.00	1.00	1.00
NMAP TCP scan	1.00	1.00	1.00
NMAP UDP SCAN	0.99	0.99	0.99
NMAP XMAS TREE SCAN	1.00	0.99	1.00
Thing Speak	0.99	0.99	0.99
Wipro bulb	1.00	1.00	1.00

XGBoost Report			
Attack Type	Precision	Recall	F1
ARP poisioning	0.98	0.99	0.99
DDOS Slowloris	0.97	1.00	0.99
DOS SYN Hping	1.00	1.00	1.00
MQTT Publish	1.00	0.99	1.00
Metasploit Brute Force SSH	0.80	0.67	0.72
NMAP FIN SCAN	1.00	0.67	0.80
NMAP OS DETECTION	1.00	1.00	1.00
NMAP TCP scan	1.00	1.00	1.00
NMAP UDP SCAN	0.99	0.98	0.98
NMAP XMAS TREE SCAN	1.00	0.99	1.00
Thing Speak	0.95	0.94	0.95
Wipro bulb	0.98	0.93	0.96

XGBoost

Applying an XGBoost model starting with the same base hyper-parameters as the AdaBoost model, 100 base estimators and a learning rate of 2.0, resulted in an accuracy of 81.57%. Similar to the AdaBoost mode, hyper-parameter tuning was performed on the model. The same grid was searched, but the addition of the gamma and subsample hyper-parameters which are specific to XGBoost. The gamma hyper-parameter determines the smallest value of loss reduction required to make a further partition of a leaf node. The subsample determines the ratio of the training instances observed by the model each iteration before developing new trees. These are both hyper-parameters aimed at reducing over-fitting in the model. To achieve quicker cross validation with the increased number of hyper-parameters, random search was chosen over the more computationally expensive grid search.

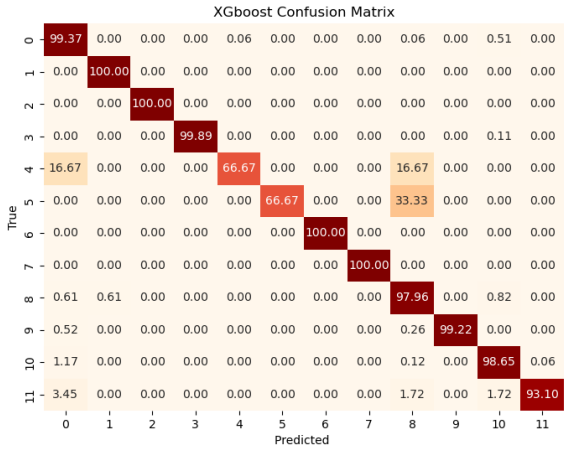


Fig. 13. XGBoost Confusion Matrix

XGBoost Randomized Search					
hyperparameter	values				
n estimators	50	100	200		
learning rate	0.1	0.5	1.0	2.0	
max depth	1.0	2.0	3.0	4.0	5.0
gamma	0.5	1.0	1.5	2.0	5.0
subsample	0.6	0.8	1.0		

The optimal model was found to have a 1.0 learning rate, 100 base estimators, a max depth of 2, 0.6 subsample, and 0.5 gamma. This best model resulted in an accuracy of 99.78%, an improvement from the 81.57% accuracy from the original XGBoost model. The macro average values for Precision, Recall and F1-Score were 97.60%, 93.46%, 95.20% respectively.

The accuracy of the AdaBoost model performed better than the XGBoost model prior to hyper-parameter tuning. Following the hyper-parameter tuning the XGBoost model outperformed the AdaBoost model in accuracy by 0.016%. That being said, given the imbalanced data set the macro averages of the Precision, Recall and F1-Score reveal that the AdaBoost model performs better at identifying each individual class, rather than just the majority class. The choice of randomized search over grid search for parameter-tuning, due to computation time, could have been a factor in the reduced performance in the XGBoost model over the AdaBoost model. While training time was not considered in the comparison of the models it is worth noting that the AdaBoost model took significantly longer to train when compared to the XGBoost model.

Comparison

Feature Relevance in Decision Tree and Random Forest Models

To gauge which features in the original dataset were most pertinent to the Decision Tree and Random Forest predictions, the 'feature_importances_' attribute containing an array of values that correspond to the importance of each feature, was accessed. Given that the algorithms were performed on PCA-transformed data, the output would indicate, not specific features, but instead, the most important principal components. Decision Tree and Random Forest shared 3 out of the top 5 most important principal components, which were PC4, PC5, and PC7.

Then, each of these PCs were investigated individually, first by determining which rows had the most significant contribution to that PC, and then by extracting the top 5 features associated with that row. For each PC, a dictionary was constructed in which 5 lists corresponding to Feature 1 thru 5 contained the feature names obtained in the previous step.

TABLE I
COMMON FEATURES ACROSS ALL THREE PRINCIPAL COMPONENTS

Feature Name
flow_SYN_flag_count

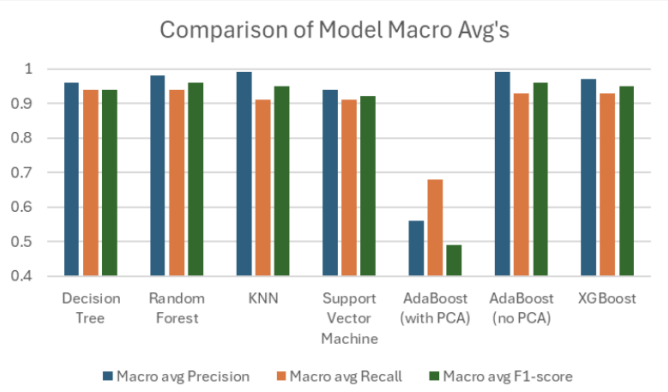
The feature that was common among PC4, PC5, and PC7 was 'flow_SYN_flag_count', which counts the number of packets in a network flow that have the SYN flag set. Given the context of this study, it could be inferred that monitoring the count of SYN flags in network flows is crucial for detecting unusual or potentially malicious activity. For example, a high number of SYN packs could indicate a SYN flood attack, which is a type of Denial of Service (DoS) attack. Recall that Fig. 1 showed that the most common attack type by far was 'DOS_SYN_Hping,' a DoS attack.

TABLE II
COMMON FEATURES ACROSS ANY TWO OF THE THREE PRINCIPAL COMPONENTS

Feature Name
bwd_bulk_rate
bwd_header_size_max
bwd_header_size_min
flow_SYN_flag_count
fwd_bulk_bytes
fwd_iat.min
payload_bytes_per_second
proto_udp
service_irc

The list encompassing the features that are shared by 2 out of the 3 top PCs are comprised of several network behavior indicators, such as 'bwd_bulk_rate' and 'fwd_bulk_bytes' which measure various aspects of data flow within the network and are crucial for understanding normal vs. abnormal behavior. Additionally, since Internet Relay Chat (IRC) is a traditional protocol for detecting botnet command and control channels, features like 'service_irc' can contribute greatly to ML algorithms designed for anomaly detection and/or classification of malicious attacks. The commonality of these features across multiple PCs indicates their robustness in contributing to the variance, making them valuable for assessing which features are most crucial in cybersecurity contexts.

Accuracy Comparison



Model	Macro avg Precision	Macro avg Recall	Macro avg F1-score
Decision Tree	0.96	0.94	0.94
Random Forest	0.98	0.94	0.96
KNN	0.99	0.91	0.95
Support Vector Machine	0.94	0.91	0.92
AdaBoost (with PCA)	0.56	0.68	0.49
AdaBoost (no PCA)	0.99	0.93	0.96
XGBoost	0.97	0.93	0.95

Fig. 14. Macro Averages for all Models

The Decision Tree model achieved a high overall accuracy but this metric was improved upon slightly by combining multiple decision trees into ensemble methods, exhibiting a 0.1% improvement. The Random Forest, XGBoost, and AdaBoost ensemble methods all had improved evaluation metrics in some respects. While the Recall for the XGboost and AdaBosot models were slightly worse, both 93%, compared to the Decision Tree models' 94%, the precision and F1-score macro average values were both improved. The most nontrivial improvement observed in the analysis was that of PCA's impact on the Boosting methods employed. The AdaBoost model performed significantly worse when the data was preprocessed using the PCA dimensionality reduction technique.

Conclusion

In this paper, the *RT-IoT2022* dataset was used to train numerous classifiers to identify different network traffic patterns, including attacking behaviors. Considering the uneven distribution of class instances, the metrics of focus for assessing the performance of the selected models were the macro averages of precision, recall, and F1-scores. These averages are particularly useful for evaluating the model's overall effectiveness across all classes, as they are not influenced by class imbalance. The models tested were Decision Tree, Random Forest, KNN, boosting methods (AdaBoost and XGBoost),

and SVM. The results demonstrate that the Decision-Tree-based models were suitable for classifying the network data in the presence of non-normal data. The ensemble methods, more specifically the Random Forest and AdaBoost models, especially after hyperparameter tuning, performed the best, albeit with relatively small improvements in the metrics of interest. The significance of the gains obtained by ensembling may be debated, especially considering the increased computational demands of ensemble methods. In practice, the choice of model would depend on factors such as urgency, computational resources, and the cost of predictive errors. Another point of consideration is whether, in the context of the cybersecurity industry, the improvements exhibited in this analysis are both statistically significant and worthwhile.

Future Considerations

Having implemented our own dimensionality reduction strategy (PCA), and reviewed existing literature, it is worth considering the viability of other ML algorithms such as K-Nearest Neighbors (KNN), whose success is demonstrated in Sharafaldin et al. [SLG18]. Additionally, the article does not use PCA analysis, but rather a RandomForestRegressor class of scikit-learn "to select the best short feature set for each attack which can be best detection features set for each attack." [SLG18] Experimenting further with hyper-parameter tuning is an option that we will likely explore.

A deep learning classification technique would be worth investigating as neural networks become more common within data science. Additionally, a comprehensive analysis investigating the robustness and applicability of the models chosen in this study would be greatly beneficial to ensure that the most meaningful information is gleaned.

In the future, detection time could be included in the overall evaluation of each of the respective models, similar to what was done in Chauhan and Atulkar [CA21]. This metric holds significance because detection time plays a critical role in evaluating detection systems, as it directly influences the ability to quickly identify and respond to network intrusions.

References

[SLG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization". In: *International Conference on Information Systems Security and Privacy*. 2018. URL: https://api.semanticscholar.org/CorpusID:4707749.

[MTC19] N. Moustafa, B. Turnbull, and K.-K. R. Choo. "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things". In: *IEEE Internet of Things Journal* 6.3 (June 2019), pp. 4815–4830. DOI: 10.1109/JIOT.2018.2871719.

[CA21] Chauhan and Atulkar. "Selection of Tree Based Ensemble Classifier for Detecting Network Attacks in IoT". In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. 2021.

[GLS21] S. Guo, Y. Liu, and Y. Su. "Comparison of Classification-based Methods for Network Traffic Anomaly Detection". In: *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. 2021, pp. 360–364. DOI: 10.1109/IMCEC51613.2021.9482274.

[Dan+22] P. K. Danso et al. "Ensemble-based Intrusion Detection for Internet of Things Devices". In: *2022 IEEE 19th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*. 2022, pp. 34–39. DOI: 10.1109/HONET56683.2022.10019140.

[SN24] B. S. and Rohini Nagapadma. *RT-IoT2022*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5P338. 2024.