Noah Pragin
05/22/2025
AI 541 ML in the Real World
Dr. Kiri Wagstaff

AI 541 Final: Identifying Malicious IoMT Network Traffic

Problem Statement

The prediction problem for this project is to classify MQTT network traffic, a lightweight messaging protocol commonly used in Internet of Things (IoT) devices, as either benign or malicious for Internet of Medical Things (IoMT) devices. The goal of this model is to deploy on-device to detect malicious traffic and terminate malicious connections. The problem I aim to address is critical because as we integrate networks into our medical devices, we introduce new attack vectors that adversaries can exploit, specifically for DOS, DDOS, and Malformed Data attacks. Consumers of medical devices rely on some devices for security, such as a baby monitor, and for contacting emergency responses, as seen in the case of a fall detector. Suppose consumers must rely on these devices for their health, and these devices must expose network interfaces to function. In that case, everyone should have peace of mind that these interfaces are secure, or at the very least, can become secure should they become compromised. The intended beneficiaries and users of this model are the engineers who deploy medical devices exposing MQTT endpoints and the consumers of these products. The engineers behind these products recognize that safety is a crucial concern when it comes to medical devices and that integrating internet connectivity introduces not only new points of failure but also potential attack vectors for adversaries. Consumers of medical devices may not be aware of these risks, let alone whether their device connects to the internet; their primary concern is that their medical device is safe to use and performs its function at all times.

Dataset

The dataset used to train the classification model, CICIoMT2024, is provided by Sajjad Dadkhah et al. from the Canadian Institute for Cybersecurity and can be accessed at www.unb.ca/cic/datasets/iomt-dataset-2024.html. The dataset contains examples of benign traffic and five types of MQTT attacks: DoS connect flood, DDoS connect flood, DoS publish flood, DDoS publish flood, and malformed data. The data was collected using fifteen medical devices, including glucometers, pulse monitors, infusion pumps, fall monitors, baby monitors, and more. We combine all benign and MQTT training and test data into a main training and test dataset, respectively. The resulting dataset comprises 557,000 examples, with 455,670 in the training set and 101,322 in the testing set. We intentionally combine all malicious traffic into a single malicious class, thereby losing information about the specific attack type, as the model's applications are agnostic to this information.

As a result, we have a relatively balanced dataset, with 58% of examples being malicious and 42% of examples being benign. However, this is not the class balance we expect to see at

deployment, where benign traffic should significantly outnumber malicious traffic. It is important to note that each entry in the dataset is an average over a window of 100 packets. Since each entry in the dataset is an average over a window of 100 packets, all features are real-valued; however, some features are limited to the range [0, 1]. The dataset provides 11 features that describe the packet size and transmission rate, 11 features that represent either the proportion or number of packets with a particular flag set, and 16 features that define the application, transport, network, and data link layer protocol used in packet transmission. In total, we use 38 features. Note that the descriptions attached to each feature below are provided by Dadkhah et al.

Packet Feature Statistics and Descriptions

name	min	max	mean	50%	Description
Header_Length	0	9.90E+06	2.29E+05	17173.2	Mean of the Header Lengths of the Transport Layer
Rate	0	1.76E+06	2985.31	9.95	Speed of packet transmission within a window in packets/sec
Tot sum	66	23467	1833.17	732.42	Total packet length within the aggregated packets (window)
Min	42	1514	116.92	66	Shortest packet length within the aggregated packets (window)
Max	43.8	1514	292.68	95.7	Longest packet length within the aggregated packets (window)
AVG	42.18	1514	174.68	75.61	Mean of the packet length within the aggregated packets (window)
Std	0	721.15	61.12	9.04	Standard deviation of the packet length within the aggregated packets (window)
Tot size	43.8	1514	174.7	75.6	(Avg.) Length of the Packet
IAT	-1.28	1.69E+08	8.47E+07	8.47E+07	Interval mean between the current and previous packet in the window
Number	1	15	9.5	9.5	Total number of packets in the window
Variance	0	1	0.84	0.9	Variance of the packet lengths in the window

Flag Feature Statistics and Descriptions

name	min	max	mean	50%	Description
fin_flag_number	0	1	0.08	0	Proportion of packets with FIN flags in the window
syn_flag_number	0	1	0.09	0	Proportion of packets with SYN flags in the window
rst_flag_number	0	1	0.09	0	Proportion of packets with RST flags in the window
psh_flag_number	0	1	0.33	0.3	Proportion of packets with PSH flags in the window
ack_flag_number	0	1	0.83	0.9	Proportion of packets with ACK flags in the window
ece_flag_number	0	0.2	0	0	Proportion of packets with ECE flags in the window
cwr_flag_number	0	0.1	0	0	Proportion of packets with CWR flags in the window
ack_count	0	11.2	0.44	0	Count of ACK flag occurrences in packets
syn_count	0	10.7	0.8	0.2	Count of SYN flag occurrences in packets
fin_count	0	151.74	0.62	0	Count of FIN flag occurrences in packets
rst_count	0	9548.5	970.61	38.5	Count of RST flag occurrences in packets

Protocol Feature Statistics and Descriptions

name	min	max	mean	50 %	Description
Protocol Type	0	17	6.35	6	Mode of protocols found in the window
HTTP	0	1	0.01	0	Average no. of HTTP packets in the window
HTTPS	0	1	0.05	0	Average no. of HTTPS packets in the window
DNS	0	1	0	0	Average no. of DNS packets in the window
Telnet	0	0	0	0	Average no. of Telnet packets in the window
SMTP	0	0	0	0	Average no. of SMTP packets in the window
SSH	0	0.1	0	0	Average no. of SSH packets in the window
IRC	0	0	0	0	Average no. of IRC packets in the window
TCP	0	1	0.96	1	Average no. of TCP packets in the window
UDP	0	1	0.04	0	Average no. of UDP packets in the window
DHCP	0	0.3	0	0	Average no. of DHCP packets in the window
ARP	0	1	0.01	0	Average no. of ARP packets in the window
ICMP	0	0.9	0	0	Average no. of ICMP packets in the window
IGMP	0	0.7	0	0	Average no. of IGMP packets in the window
IPv	0	1	0.99	1	Average no. of IPv packets in the window
LLC	0	1	0.99	1	Average no. of LLC packets in the window

Models

I employ seven different models to comprehensively evaluate the performance of techniques for addressing the dataset's challenges across model types. I use two logistic regression models, one with a degree of one and another with a degree of three, to establish baseline performance and assess whether inter-feature relationships are important for this classification problem. SKLearn does not provide a degree parameter, so we transform our features using a PolynomialFeatures object. I employ two ensemble methods: a Random Forest, chosen for its generalizability, and an AdaBoost Forest, selected to compare how bagging and boosting differ in their response to challenge mitigation techniques. I employ two Support Vector Machine variants, one with an RBF kernel to capture highly complex decision boundaries and another with a polynomial degree 3 kernel, to determine whether the benefits of kernel methods stem from the specific kernel choice or the SVM framework itself. Finally, I included a neural network as a highly expressive model capable of learning intricate feature interactions with an increased risk of overfitting. I describe the hyperparameters in the table below.

Model Name	Hyperparameters	Hyperparameter Reasoning
Logistic Regression D1	penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, solver='lbfgs', max_iter=100,	Regularization was found to not need to be increased, no other hyperparameters were tuned
Logistic Regression D3	penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, solver='lbfgs', max_iter=100 PolynomialFeatures Parameters: degree=3, interaction_only=False, include_bias=True	Regularization was found to not need to be increased, no other hyperparameters were tuned
Random Forest	n_estimators=200, criterion='gini', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='log2', min_impurity_decrease=0.0	I increased the number of estimators and switched to a lower number of max_features because I suspect some of the features are significantly less informative than others, so I wanted to increase the chances of filtering out random noise
AdaBoost Forest	estimator=DecisionTreeClassifier, n_estimators=50, learning_rate=1.0	The default learning rate was found to perform very well
SVM - RBF	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=False, max_iter=1000	Increasing regularization was not found to be helpful

SVM - Poly D3	C=1.0, kernel='poly', degree=3, gamma='scale', coef0=0.0, shrinking=True, max_iter=1000	Increasing regularization was not found to be helpful
MLP	hidden_layer_sizes=(64,64, 64), solver='adam', alpha=0.0001, shuffle=True, beta_1=0.9, beta_2=0.999, epsilon=1e−08, momentum=0.9, early_stopping=false, learning_rate_init=0.001, max_iter=25, learning_rate='constant', activation='relu'	Enough hidden layers to be a universal function approximator, plus one for good measure, early stopping for termination criteria.
Dummy - Most Frequent	strategy='most_frequent'	Baseline
Dummy - Stratified	strategy='stratified'	Baseline

Evaluation

I use the dedicated test set to generate the evaluation metrics. I don't want to use class-stratified methods because the dataset's class imbalance does not reflect the class balance at test time. I would have liked to perform group cross-validation, using a device identifier as the group, to measure generalization to new devices better. However, the dataset curators did not include that data. I will have two baseline methods: a majority class classifier that will always classify traffic as malicious and a stratified classifier. I report accuracy, recall, precision, and F1 score to provide a comprehensive view of model performance, as misclassification errors depend on the device on which the model is deployed. For example, a heart pump will incur significantly higher costs due to false positives (falsely identifying a benign connection as malicious) than a baby monitor.

Challenges

1. Feature Scaling

In the dataset, about half of the features are real numbers, while the other half are rates bounded to the interval [0, 1]. Some of the models I've selected are robust to the varying scale of the features, such as the tree-based methods. However, most of my models are not, such as SVM, DNN, and regularized logistic regression. To mitigate the risks of weighing real-numbered features more importantly and penalizing them more in regularized modes, I will investigate different methods of scaling the features.

A. Strategy 1: Normalize

Normalization, also known as min-max scaling, is the process of transforming your data to the range [0, 1]. For each feature, we set the maximum value in the dataset to 1, the minimum value to 0, and scale every value in between based on its closeness to the minimum or maximum value. You can normalize a data point by subtracting the minimum value from it and then dividing the result by the difference between the minimum and maximum values.

B. Strategy 2: Normalize with 5% Winsorization

Winsorization is a method to reduce outliers in a dataset. Winsorization is parameterized by a percentile, with all values within that percentile (0-5% and 95-100%) being clipped to the 5th and 95th percentile values, respectively. Features such as Rate and Header Length suffer from huge outliers; winsorization will mitigate the risk of normalization generating small decimal values for a majority of our data points. We apply winsorization before normalization for this strategy.

C. Strategy 3: Standardize

Standardization is another method of bringing features of different scales to a similar one. After standardization, the feature will have a mean of 0 and a standard deviation of 1 across the entire dataset. To standardize a dataset, you must gather the mean and standard deviation of each feature. Then, for each feature, subtract the mean and divide by the standard deviation.

D. Strategy 4: Standardization with 5% Winsorization

Similar to Normalization, Standardization risks generating very small values for features with outliers of a large magnitude. For this reason, we also explore standardization with winsorization, like strategy 2, to mitigate these risks. We apply winsorization before standardization for this strategy.

E. Do Nothing

All models selected can predict a class if the features are not normalized or standardized. This method will provide a baseline to assess the effectiveness of standardization and normalization, as well as the benefits of winsorization.

2. Feature Reduction

I have 38 features in the dataset, but statistics vary significantly across these features. Some features exhibit a very high skew toward zero, containing predominantly zero values. I aim to investigate whether these highly skewed features provide meaningful information for classification, whether they can be removed, or if they can be combined into fewer features using Principal Component Analysis (PCA) to reduce dimensionality. Most of the features represent one-hot encodings of the application, transport, or network layer protocols used in packet transmission. However, some features are packet flag counts, while others represent packet flag rates, creating potential redundancy. For all experiments addressing feature reduction, we first normalize all features.

A. Strategy 1: Do Nothing

All models selected can predict a class regardless of the data's dimensionality. This method will provide a baseline to see the effectiveness of the proceeding strategies.

B. Strategy 2: PCA

PCA is a method of reducing the dimensionality of a dataset. PCA works by transforming the points to a new coordinate system, which preserves the maximum variance of the data. PCA works by identifying the principal components of the features by finding the eigenvectors and values of the covariance matrix and then selecting the N eigenvectors that correspond to the largest eigenvalues. I will use PCA to combine all transport layer protocols, all data link layer protocols, all network layer protocols, and all application layer protocols, significantly reducing the dimensionality of the protocol-related features from 16 to 5. This reduction to four features aligns with the ISO and IP/TCP network model hierarchy, allowing each principal component to capture the variance within its respective network layer. I do not apply PCA to the "Protocol Type" feature. I will also apply PCA to all flag-related features, combining any features that have both a rate and a count feature into a single feature. This will reduce the flag-related features from 11 to 7, as we can combine four sets of two features into one feature each.

C. Strategy 3: Recursive Feature Elimination (Backward Selection)

Recursive Feature Elimination works by leveraging a feature importance measure from a model, such as the coefficients of a linear model or the feature importance of a random forest. Recursive Feature Elimination is a greedy operation where a model is fit to the dataset using all features, then, using the measure of feature importance, the least important feature is removed. This process is repeated until the desired number of features or some error tolerance is reached. For our experiments, we will run RFE until one feature remains, generating a comprehensive view of how performance changes as features are removed. We will use the Random Forest described in the Models section as the feature importance estimator.

Challenge 3: Class Imbalance

My dataset exhibits a class imbalance (Benign/Malicious) of 41%/59% across both datasets, with 42%/58% in the training set and 37%/63% in the test set. This is especially problematic because the deployment setting expects most network traffic to be benign. However, false negatives are significantly more costly than false positives, so the heavy representation of malicious data is beneficial in reducing the risk of low recall.

A. Strategy 1: Undersample the Majority Class

Undersampling the majority class is a method to artificially balance your classes; however, it should be used carefully because it will affect the posterior probabilities of your model. Further, undersampling the majority class could lead to adverse effects if the amount of data you have is not enough to learn the underlying function after undersampling. To undersample the majority class, you simply remove data points that have the majority label. After undersampling the majority class, we will randomly remove 70,206 examples of malicious data, resulting in a total dataset size of 385,464.

B. Strategy 2: Oversampling the Minority Class

Oversampling the minority class is a method to artificially balance your classes; however, like undersampling the majority class, it should be used carefully because it will affect the posterior probabilities of your model. Something to note is that oversampling the minority class will not help performance if you do not have enough examples to learn the underlying function,

as you will suffer from underfitting. To oversample the minority class, you simply duplicate data points that have the minority label. After undersampling the majority class, we will randomly duplicate 70,206 examples of benign data, resulting in a total dataset size of 525,876.

C. Strategy 3: Class Weighted Learning

Class weighting is the process of assigning different misclassification costs to examples based on their prevalence in the dataset. Class-weighted learning can be implemented by modifying the loss function of your model to increase the cost of misclassifying the minority class by a factor that reflects its relative size compared to the majority class. For tree-based models that do not optimize a loss function, we can multiply the entropy calculation for the minority class by the same factor. We will only run this strategy on classifiers that support the class_weight parameter in SKLearn, which we will set to "balanced".

D. Strategy 4: Do Nothing

All selected models can predict a class regardless of class imbalance. This method will provide a baseline to contrast the effectiveness of undersampling and oversampling the dataset.

Results

F = F1 Score; P = Precision; R = Recall; A = Accuracy; where Malicious is the positive class.

Results were generated using the test set.

Feature Scaling

Model	Do Nothing	Normalized	Normalized Winsorized	Standardized	Standardized Winsorized
Dummy - Most Frequent	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88
Dummy - Stratified	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95
Logistic Regression D1	F: 85.50 P: 86.06 R: 85.85 A: 85.85	F: 97.83 P: 97.85 R: 97.82 A: 97.82	F: 94.83 P: 95.17 R: 94.78 A: 94.78	**F: 97.84** **P: 97.86** **R: 97.84** **A: 97.84**	F: 94.83 P: 95.18 R: 94.78 A: 94.78
Logistic Regression D3	F: 38.72 P: 72.00 R: 46.34 A: 46.34	**F: 98.98** **P: 99.00** **R: 98.98** **A: 98.98**	F: 94.07 P: 94.22 R: 94.03 A: 94.03	F: 98.92 P: 98.95 R: 98.92 A: 98.92	F: 98.31 P: 98.35 R: 98.31 A: 98.31
Random Forest	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	F: 99.58 P: 99.59 R: 99.58 A: 99.58	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	F: 99.57 P: 99.58 R: 99.57 A: 99.57
AdaBoost Forest	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.18 P: 99.19 R: 99.18 A: 99.18	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.19 P: 99.20 R: 99.19 A: 99.19
SVM - RBF	F: 98.27 P: 98.34 R: 98.26 A: 98.26	**F: 98.9**9 P: 99.01 **R: 98.99** **A: 98.99**	F: 95.49 P: 95.94 R: 95.44 A: 95.44	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	F: 95.35 P: 95.83 R: 95.30 A: 95.30
SVM - Poly D3	F: 13.02 P: 11.34 R: 19.42 A: 19.42	**F: 98.99** **P: 99.01** **R: 98.99** **A: 98.99**	F: 93.40 P: 93.58 R: 93.48 A: 93.48	F: 70.26 P: 81.09 R: 74.56 A: 74.56	F: 98.40 P: 98.43 R: 98.39 A: 98.39
MLP	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 98.98 P: 99.00 R: 98.98 A: 98.98	F: 98.57 P: 98.61 R: 98.56 A: 98.56	**F: 99.09** **P: 99.11** **R: 99.09** **A: 99.09**	F: 95.80 P: 96.18 R: 95.76 A: 95.76

Standardization and normalization proved helpful for dealing with feature scaling across almost all models except for the tree-based methods. This discrepancy was expected, as tree-based methods, such as the AdaBoost Classifier and Random Forest Classifier, are robust to

varying feature scales. Furthermore, this property foreshadowed the harm of winsorization on model performance, as we are needlessly clamping potentially informative values. Standardization generally outperformed normalization by a marginal difference across most models. Surprisingly, winsorization hurt performance across all models, indicating that values I perceived as outliers were highly informative to the classification task.

PCA – Protocols

Note: PCA – Layer 2 means we combine the features corresponding to Layer 2 protocols into a single feature, etc. PCA – Protocols means we do all protocol-specific PCA approaches.

	Do Nothing	PCA - Layer 2	PCA - Layer 3	PCA - Layer 4	PCA - Layer 7	PCA - Protocols
Dummy - Most Frequent	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88
Dummy - Stratified	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95
Logistic Regression - D1	**F: 97.83** **P: 97.85** **R: 97.82** **A: 97.82**	F: 97.82 P: 97.84 R: 97.82 A: 97.82	**F: 97.83** **P: 97.85** **R: 97.82** **A: 97.82**	F: 97.82 P: 97.84 R: 97.82 A: 97.82	F: 95.04 P: 95.05 R: 95.06 A: 95.06	F: 94.98 P: 95.00 R: 95.00 A: 95.00
Logistic Regression - D3	F: 98.98 P: 99.00 R: 98.98 A: 98.98	F: 98.98 P: 99.00 R: 98.97 A: 98.97	**F: 99.00** P: 99.02 **R: 99.00** **A: 99.00**	**F: 99.00** **P: 99.03** **R: 99.00** **A: 99.00**	F: 98.97 P: 98.99 R: 98.97 A: 98.97	F: 98.99 P: 99.01 R: 98.98 A: 98.98
Random Forest	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**	F: 99.74 P: 99.74 R: 99.74 A: 99.74	**F: 99.75** **P: 99.75** **R: 99.75** **A: 99.75**
AdaBoost	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.22 P: 99.23 R: 99.22 A: 99.22	F: 99.22 P: 99.23 R: 99.22 A: 99.22
SVM - RBF	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	F: 98.96 P: 98.99 R: 98.96 A: 98.96	F: 98.96 P: 98.99 R: 98.96 A: 98.96
SVM - Polynomial D3	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	F: 98.98 P: 99.01 R: 98.98 A: 98.98	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	F: 98.97 P: 98.99 R: 98.96 A: 98.96	F: 98.98 P: 99.00 R: 98.98 A: 98.98
MLP	F: 98.98 P: 99.00 R: 98.98 A: 98.98	**F: 99.16** **P: 99.17** **R: 99.16** **A: 99.16**	F: 99.07 P: 99.09 R: 99.07 A: 99.07	F: 99.12 P: 99.13 R: 99.11 A: 99.11	F: 98.98 P: 99.00 R: 98.97 A: 98.97	F: 99.00 P: 99.02 R: 99.00 A: 99.00

PCA – Flags

Note: PCA – SYN Flag means we combine both features encoding SYN Flag frequency into a single feature, and so on. PCA – Flags means we do all flag-specific PCA approaches. PCA – All means we do both PCA – Flags and PCA – Protocols.

	Do Nothing	PCA - Flags	PCA - SYN Flag	PCA - FIN Flag	PCA - RST Flag	PCA - ACK Flag	PCA - All
Dummy - Most Frequent	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88	F: 48.55 P: 62.88 R: 62.88 A: 62.88
Dummy - Stratified	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95	F: 52.48 P: 53.28 R: 51.95 A: 51.95
Logistic Regression - D1	F: 97.83 P: 97.85 R: 97.82 A: 97.82	**F: 97.92** **P: 97.94** **R: 97.92** **A: 97.92**	F: 97.90 P: 97.91 R: 97.89 A: 97.89	F: 97.85 P: 97.87 R: 97.84 A: 97.84	F: 97.82 P: 97.84 R: 97.82 A: 97.82	F: 97.82 P: 97.84 R: 97.81 A: 97.81	F: 95.27 P: 95.28 R: 95.28 A: 95.28
Logistic Regression - D3	F: 98.98 P: 99.00 R: 98.98 A: 98.98	F: 98.96 P: 98.99 R: 98.96 A: 98.96	F: 98.98 P: 99.00 R: 98.97 A: 98.97	F: 98.99 **P: 99.02** R: 98.99 A: 98.99	F: 98.97 P: 98.99 R: 98.97 A: 98.97	**F: 99.00** **P: 99.02** **R: 99.00** **A: 99.00**	F: 98.95 P: 98.97 R: 98.94 A: 98.94
Random Forest	F: 99.75 P: 99.75 R: 99.75 A: 99.75	**F: 99.77** **P: 99.77** **R: 99.77** **A: 99.77**	F: 99.75 P: 99.75 R: 99.75 A: 99.75	F: 99.75 P: 99.76 R: 99.75 A: 99.75	F: 99.75 P: 99.75 R: 99.75 A: 99.75	F: 99.75 P: 99.75 R: 99.75 A: 99.75	F: 99.76 P: 99.76 R: 99.76 A: 99.76
AdaBoost	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.15 P: 99.16 R: 99.15 A: 99.15	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.15 P: 99.15 R: 99.15 A: 99.15	F: 99.23 P: 99.24 R: 99.23 A: 99.23	**F: 99.25** **P: 99.26** **R: 99.25** **A: 99.25**	F: 99.15 P: 99.16 R: 99.15 A: 99.15
SVM - RBF	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	**F: 98.99** P: 99.01 R: 98.98 A: 98.98	F: 98.98 P: 99.01 R: 98.98 A: 98.98	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	F: 98.96 P: 98.99 R: 98.96 A: 98.96
SVM - Polynomial D3	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	F: 98.98 P: 99.01 R: 98.98 A: 98.98	**F: 98.99** P: 99.01 R: 98.98 A: 98.98	**F: 98.99** P: 99.01 R: 98.98 A: 98.98	**F: 98.99** **P: 99.02** **R: 98.99** **A: 98.99**	**F: 98.99** P: 99.01 **R: 98.99** **A: 98.99**	F: 98.96 P: 98.99 R: 98.96 A: 98.96
MLP	F: 98.98 P: 99.00 R: 98.98 A: 98.98	**F: 99.16** **P: 99.17** **R: 99.16** **A: 99.16**	F: 99.13 P: 99.14 R: 99.13 A: 99.13	F: 99.15 P: 99.16 R: 99.15 A: 99.15	F: 99.11 P: 99.12 R: 99.10 A: 99.10	F: 99.13 P: 99.15 R: 99.13 A: 99.13	F: 99.09 P: 99.11 R: 99.09 A: 99.09

Feature Reduction

Model	Do Nothing	Best PCA	Recursive Feature Elimination
Dummy - Most Frequent	A: 62.88 @ 38 features	A: 62.88 @ 23 features	A: 62.88 @ 1 feature
Dummy - Stratified	A: 51.95 @ 38 features	A: 51.95 @ 23 features	A: 51.95 @ 1 feature
Logistic Regression D1	A: 97.82 @ 38 features	**A: 97.92** **@ 34 features**	A: 97.83 @ 31 features
Logistic Regression D3	A: 98.98 @ 38 features	**A: 99.00** **@ 36 features**	A: 98.98 @ 23 features
Random Forest	A: 99.75 @ 38 features	**A: 99.77** **@ 34 features**	A: 99.76 @ 25 features
AdaBoost Forest	A: 99.25 @ 38 features	A: 99.25 @ 36 features	**A: 99.30** **@ 14 features**
SVM - RBF	A: 98.99 @ 38 features	A: 98.99 @ 36 features	**A: 98.99** **@ 25 features**
SVM - Poly D3	A: 98.99 @ 38 features	A: 98.99 @ 37 features	**A: 98.99** **@ 27 features**
MLP	A: 98.98 @ 38 features	A: 99.16 @ 34 features	**A: 99.17** **@ 30 features**

For the feature selection challenge, I attempted PCA in a targeted manner. Specifically, I combined perfectly multicollinear features (the protocol features) and redundant features (the flag features, which recorded both a proportion and a count). Both of these approaches generally harmed model performance, except in the case of the Random Forest, MLP, and D1 Logistic Regression, which had a marginal performance increase. Performance was generally maintained across all models, except for the AdaBoost Classifier, when running PCA – Flags, where redundant features were reduced. However, performance was generally harmed when combining perfectly multicollinear features, as observed when running PCA – Protocols and PCA – All. Recursive Feature Elimination was highly effective in estimating feature importance and removing uninformative features in a manner that minimized performance degradation. Across all models except the Random Forest and D1 Logistic Regression, performance increased or decreased by very little while reducing the feature set from 38 to [14, 31]. I would strongly recommend using Recursive or Sequential Feature Elimination/Selection in future projects where the dataset is suspected to have uninformative features. The report's appendix contains figures that plot test performance over the feature elimination steps for each model.

Class Imbalance

Model	Do Nothing	Oversample	Undersample	Class-Weighted Learning
Dummy - Most Frequent	**F: 48.55** **P: 62.88** **R: 62.88** **A: 62.88**	F: 20.09 P: 37.12 R: 37.12 A: 37.12	F: 20.09 P: 37.12 R: 37.12 A: 37.12	—
Dummy - Stratified	**F: 52.48** **P: 53.28** **R: 51.95** **A: 51.95**	F: 50.94 P: 53.43 R: 50.09 A: 50.09	F: 50.94 P: 53.43 R: 50.09 A: 50.09	—
Logistic Regression D1	**F: 97.85** **P: 97.86** **R: 97.84** **A: 97.84**	F: 97.74 P: 97.77 R: 97.73 A: 97.73	F: 97.75 P: 97.78 R: 97.74 A: 97.74	F: 97.74 P: 97.77 R: 97.73 A: 97.73
Logistic Regression D3	**F: 99.07** **P: 99.09** **R: 99.07** **A: 99.07**	F: 99.01 P: 99.03 R: 99.00 A: 99.00	F: 98.99 P: 99.01 R: 98.98 A: 98.98	F: 99.00 P: 99.02 R: 99.00 A: 99.00
Random Forest	**F: 99.77** **P: 99.77** **R: 99.77** **A: 99.77**	**F: 99.77** **P: 99.77** **R: 99.77** **A: 99.77**	F: 99.72 P: 99.73 R: 99.72 A: 99.72	**F: 99.77** **P: 99.77** **R: 99.77** **A: 99.77**
AdaBoost Forest	F: 95.53 P: 95.84 R: 95.59 A: 95.59	F: 95.48 P: 95.79 R: 95.54 A: 95.54	**F: 95.90** **P: 96.14** **R: 95.95** **A: 95.95**	—
SVM - RBF	**F: 98.98** **P: 99.00** **R: 98.98** **A: 98.98**	F: 98.95 P: 98.98 R: 98.95 A: 98.95	F: 98.96 P: 98.98 R: 98.95 A: 98.95	F: 98.96 P: 98.98 R: 98.95 A: 98.95
SVM - Poly D3	**F: 98.99** **P: 99.01** **R: 98.99** **A: 98.99**	F: 98.96 P: 98.98 R: 98.96 A: 98.96	F: 98.96 P: 98.98 R: 98.96 A: 98.96	F: 98.96 P: 98.99 R: 98.96 A: 98.96
MLP	F: 98.92 P: 98.95 R: 98.92 A: 98.92	**F: 98.99** **P: 99.01** **R: 98.99** **A: 98.99**	F: 98.98 P: 99.00 R: 98.97 A: 98.97	—

Across all models except the MLP Classifier and the AdaBoost Forest, doing nothing was the most effective approach to dealing with class imbalance. This is likely because both classes are well-represented in the training dataset, despite the imbalance, allowing the models to learn the underlying function effectively. I am surprised that this is true without any classification threshold tuning. However, I suspected this would be the case when accuracy, precision, and recall numbers aligned closely throughout all experiments across all models, indicating the model did not have a bias toward predicting one class more than the other. I suspect that the

MLP Classifier performed poorly when ignoring the class imbalance in the dataset due to chance and increased training time. The MLP Classifier is trained over a fixed number of epochs, so increasing the number of examples in the training set when oversampling allows the model to update its weights more times. I believe the MLP Classifier performed better when undersampling the dataset because the reduction in the number of weight updates was not enough to result in underfitting, and the model settled in a better area of the optimization space. I am unsure why the AdaBoost Classifier performs better than the baseline when undersampling the dataset but worse than the baseline when oversampling the dataset. Initially, I considered the possibility that boosted forests might be more sensitive to class imbalance; however, this does not address the discrepancy between the performance of oversampling and undersampling. I suspect that when undersampling, we inadvertently discarded noisy samples, allowing the boosted forest to ignore examples that were potentially very difficult to separate.

Reflection

I was most surprised by the class imbalance results, which showed that I was better off not addressing the class imbalance in my dataset than taking any action. This is likely due to the large dataset size and the small number of unique attacks that comprise the dataset, allowing the models to fit the underlying function well, regardless of the dataset imbalance. I was also surprised that a targeted approach to PCA performed poorly. However, after further thought, I am not surprised that the perfectly multicollinear features did not maintain their information when combined into one feature, considering that perfect multicollinearity means you can drop one feature from the set without losing any information. I was also surprised to see how well performance was maintained throughout the recursive feature elimination process. This highlights that feature elimination is a great way to balance dimensionality and performance, and I will be eager to try it next time I need to reduce my feature set.

I was unable to find someone from the class to show my results to.

I had to make several changes to my initial plans. First, I expected to have a clear-cut evaluation metric that would tell me what method was the best for my problem. However, after some careful thought, I realized that the cost matrix depends on the device on which the threat detection model is deployed. So, instead of picking a single evaluation metric, I decided to report accuracy, precision, recall, and F1 score. Another thing I had to change late into the process was normalizing my dataset before addressing challenges 1 and 2. After taking a closer look at my initial results, I realized that the performance of my models, and thus the strength of my conclusions, was poorer than I expected. My final and most disappointing change was being unable to generate results for sequential feature selection and pruning using performance on a validation set as my metric for feature importance. The code is available. However, the experiments will take several weeks to run.

If I had more time, I would like to explore several more challenges with this dataset. First, I would like to contact the dataset administrators and see if a device identifier can be incorporated into the CSV files somehow from the PCAP files they provide. This would enable me to explore

how well the models can generalize to new devices. I would also like to explore addressing label shift, as the distribution of benign vs. malicious examples is significantly different in the provided dataset than at deployment, where we expect benign network traffic to dominate by a substantial margin. I would also like to explore integrating constraints about the computational footprint of the model to determine a definitive answer as to what model I would deploy on a device parameterized by its CPU and memory.

Response to Feedback

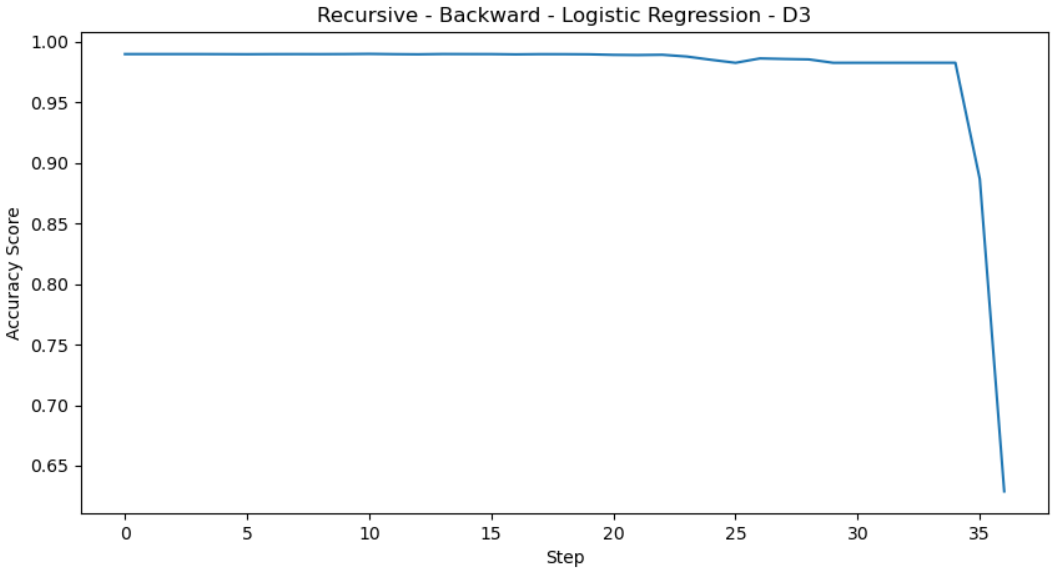
Feedback

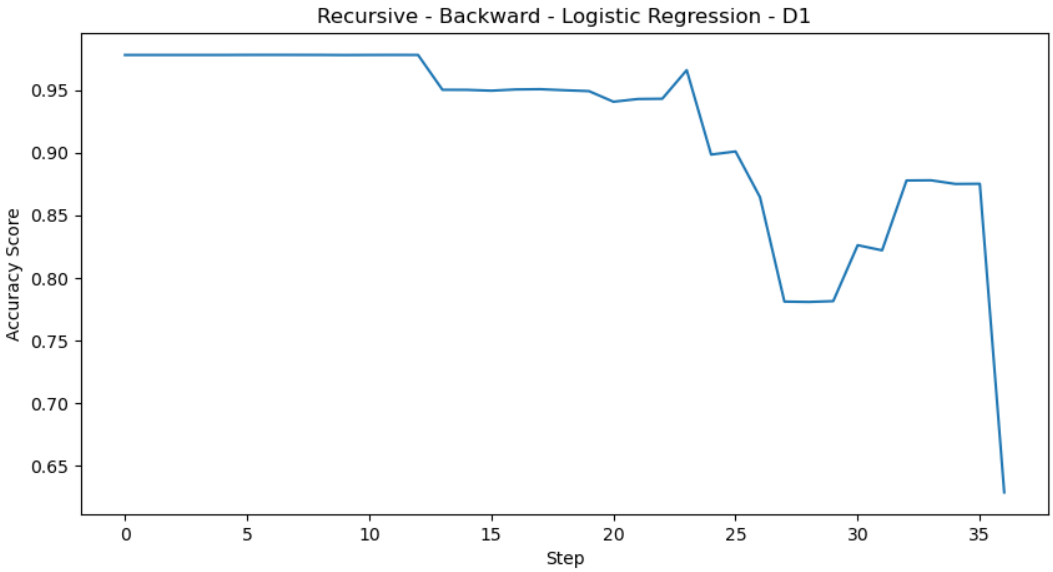
1. Consider only using exponential notation for values greater than 1e5, as this might be more legible (and fit better). You can omit the 25% and 75% percentiles and std as well, unless there's some interesting aspect you want to discuss.
    a. I've incorporated this feedback into my dataset statistics by replacing the three columns with a description column
2. Explain more about the features - what is rate, tot sum (sum of what), min/max/avg/std (of what), IAT, number (is this an index? if so, you probably don't want to use it), and variance (of what). Explain what the flags are.
    a. I've added a description column to the dataset statistics tables
3. It looks like some of the protocol features are the same for every item (Telnet, SMTP, IRC. Consider omitting these since they are not informative.
    a. I decided to continue using these features in my experiments because they could be informative if we supplemented our dataset in the future, for example via continual learning
4. Report the total number of features you used.
    a. I've added a sentence at the end of my dataset section addressing this.
5. sklearn's Logistic Regression classifier doesn't have a "degree" parameter. Explain more about what these two models are.
    a. Added a disclaimer that a feature transformation object was used to the paragraph describing the models.
6. Report key hyperparameter values even when defaults are used (LR: penalty, C, and solver).
    a. Improved reporting of Logistic Regression hyperparameters
7. Ensure you only pick hyperparameter values without looking at the test set. E.g. your comment "... the default learning rate was found to perform very well" should be based only on training (or validation) set performance, so that information from the test set does not "leak" into your training decisions.
    a. I did not use the test set to pick hyperparameters. The only time I used the test set was to evaluate the methods and for EDA.

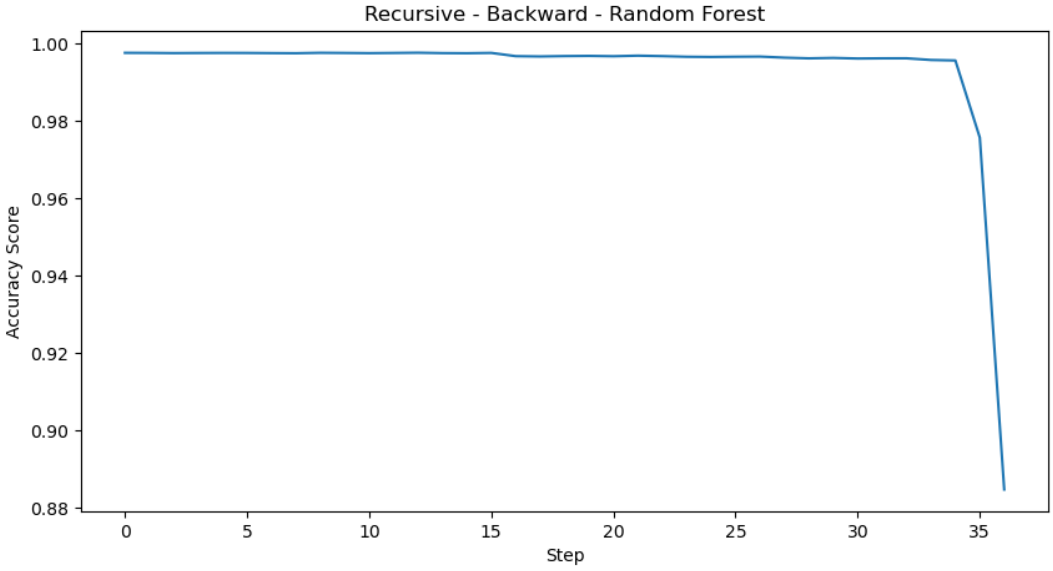
8. Since there is a dedicated test set, you don't need to do CV on the training set, unless you are using it to select hyperparameters, in which case I recommend a single train/validation split rather than k-fold CV.
    a. I removed mentions of cross validation from my evaluation section, but left in the note that I would have liked to explore device generalization by splitting data using a group identifier.
9. Update this evaluation plan once you settle on your final plans (remove language like "could" do something in favor of what you actually did).
    a. The evaluation paragraph was updated.
10. Clarify that the normalize/standardize with winsorization strategies apply winsorization *first* and then normalize/standardize. You may want to change how you title these strategies to make the order of operations clear.
    a. Added a sentence to the two strategies using winsorization to note that winsorization is applied before normalization/standardization
11. PCA: Explain why you chose 4 PCA features to represent the protocol features. Also, your feature table shows 16 protocol features but the text here says 15 - is there one you are not using? Clarify how many features the flag features will be reduced to with PCA (I couldn't tell).
    a. Added a sentence clarifying the motivation for reducing 15 of the 16 protocol features to 4 and clarified that I would not perform PCA on the Protocol Type feature. Also, added a sentence clarifying the number of features remaining after flag PCA.
12. Explain more about how you will do backward selection. You need to carve out a validation set from the training set to assess "least important feature" (cannot use the test set).
    a. Clarified Backward Selection strategy, made it clear that we are doing Recursive Feature Elimination down to one feature and using the Random Forest as the feature importance estimator
13. Class imbalance: It looks like you may have label shift going on too!
    a. Yes, and I expect even more intense label shift at deployment. If I had a do-over, I would have liked to address this challenge.
14. Explain how many items will be removed via under-sampling, and how many will be added via over-sampling.
    a. Added a sentence to the end of both the oversampling and undersampling sections
15. Explain what you will set class_weight to.
    a. Added a sentence explaining the strategy would only be run for SKLearn classifiers that support the class_weight parameter and that it will be set to "balanced"
16. Use bold to highlight the best strategy for each metric.
    a. Used bold to highlight the best strategy per model for each metric
17. Clarify whether these are CV results on the training set or test results from training a single model over the training set.

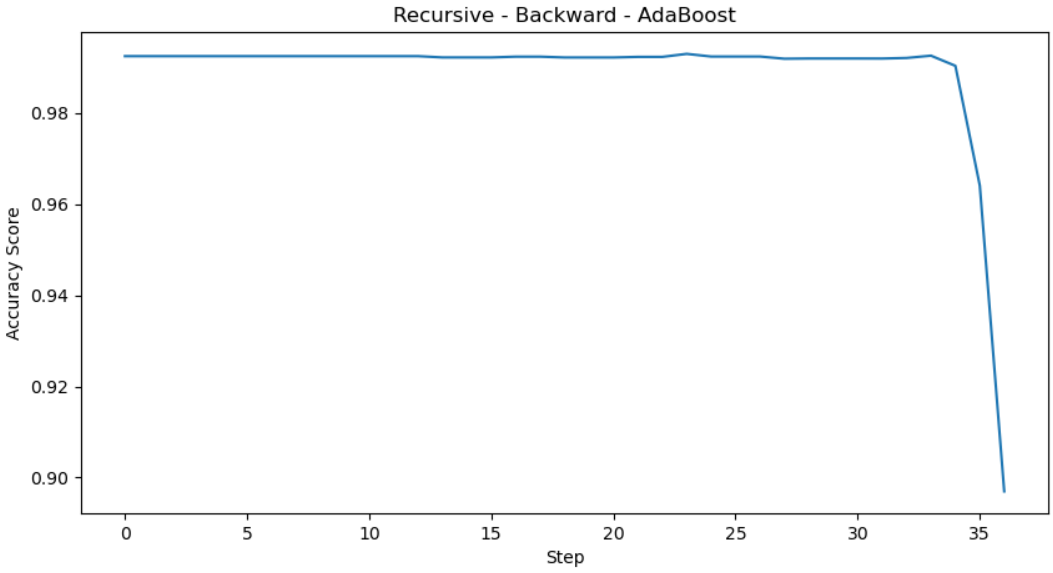
    a. Added a sentence at the beginning of the results section to clarify this
18. Placeholders for challenges 2 and 3 - flesh these out for the final report. I think "Backward iteration" is meant to be a "backward selection". The strategy titles for the class imbalance table need to be updated.
    a. Changed table header to Recursive Feature Elimination

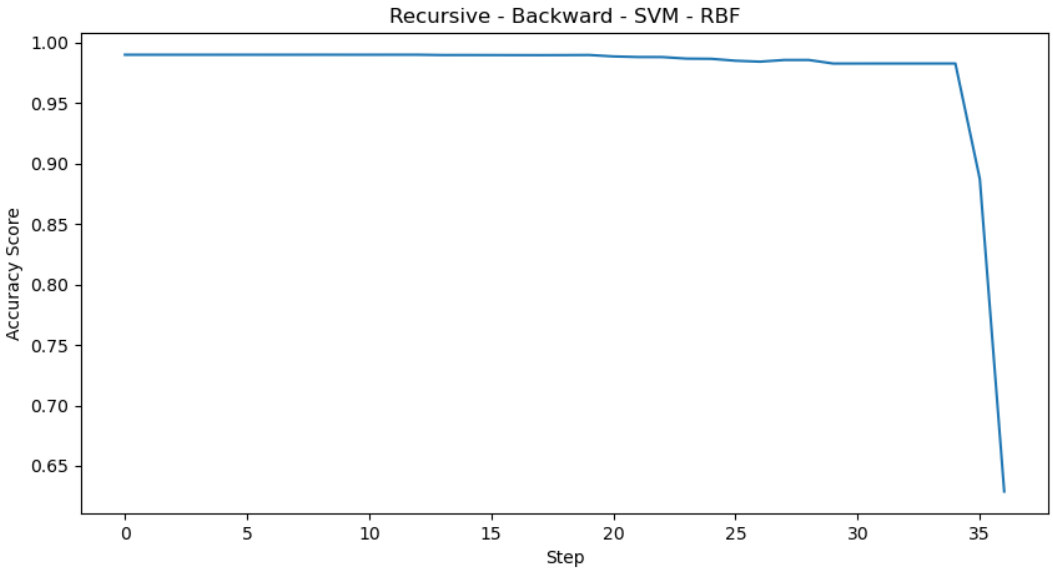
Appendix

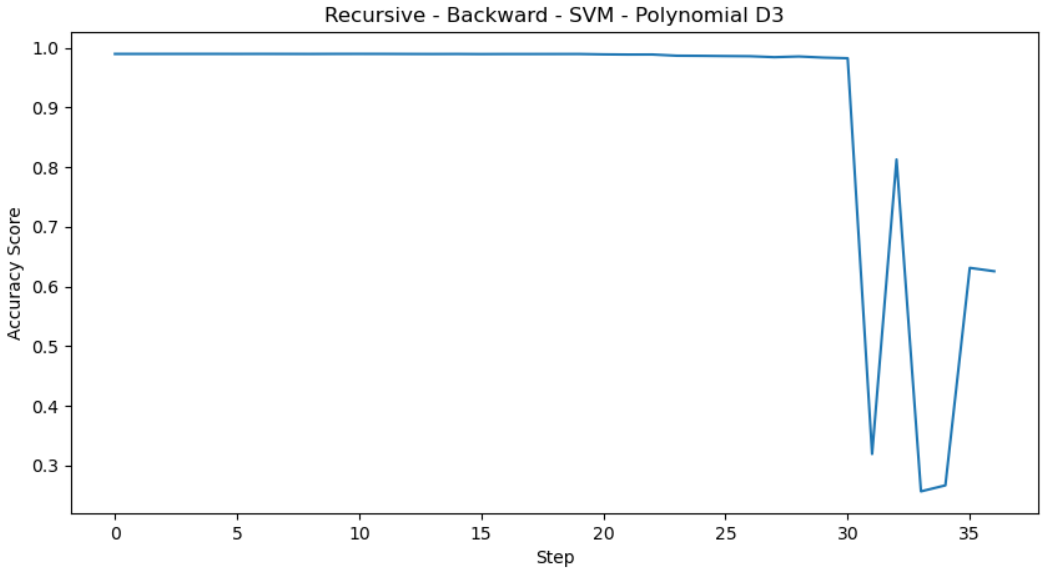


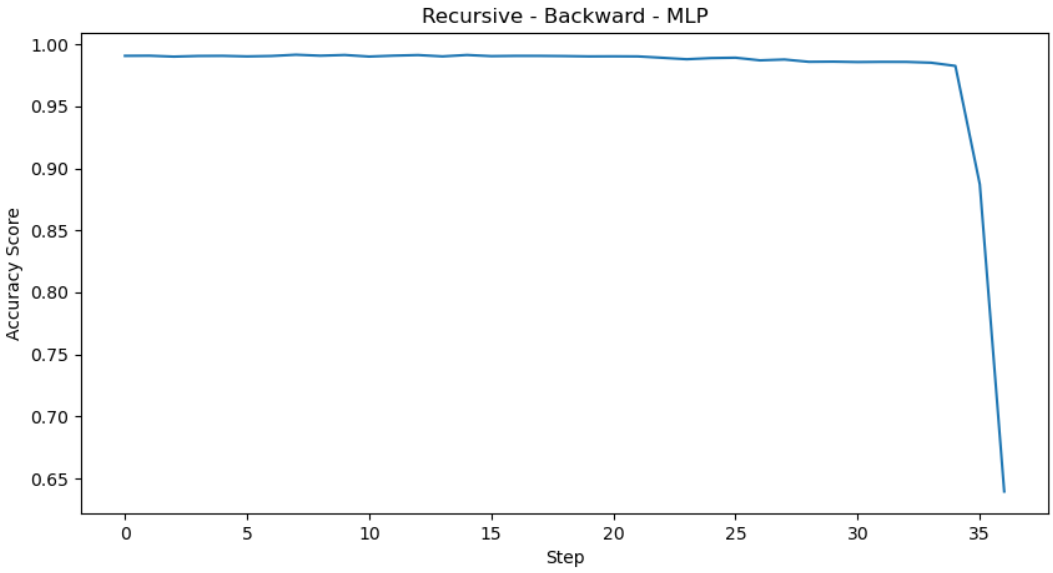



Recursive - Backward - Random Forest
Accuracy Score
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
5
10
15
20
25
30
35
Step


Recursive - Backward - AdaBoost
Accuracy Score
0.98
0.96
0.94
0.92
0.90
0
5
10
15
20
25
30
35
Step


Recursive - Backward - SVM - RBF
Accuracy Score
Step
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
5
10
15
20
25
30
35


Recursive - Backward - SVM - Polynomial D3
Accuracy Score
Step
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
5
10
15
20
25
30
35


Recursive - Backward - MLP
Accuracy Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
5
10
15
20
25
30
35
Step