- Kitsune Network Attack Dataset - Modeling Classification of ARP MitM Ettercap Attacks

Professor Sirkeci

San Jose State University

EE257

Nicholas Nuti

Partner: Hai Le

a) Dataset Description

The Kitsune Network Attack Dataset, by University of California Irvine, contains "Multivariate", "Sequential", and "Time-series" cybersecurity data for use in classification, clustering, and casual-discovery. The dataset is used to detect network attacks by possibly recognizing nine different network attacks. This report and the included modeling focuses on the ARP MitM Ettercap LAN attack, and this attack is notorious for collecting hardwire network traffic data. Attributes of the dataset involve: time-involved statistics that give insight on the sender of the packet, and recent traffic sender and receiver information of the packet. The dataset contains 2.5 million rows of input packet data with the following input features: source MAC and IP address, source IP, source and destination IP's, and source and destination of TCP/UDP socket, and each variable can be captured in 5 possible time windows. The output variable consists of an overall classification of the packet with a “1” meaning the packet was malicious and a “0” meaning the packet was safe. The dataset contains 32 bit data which may increase dependent on the time frame. There is no missing data in the dataset due to constant network traffic.

b) Dataset Visualization

The ARP MitM Ettercap sub-dataset resides within the UCI Kitsune Network Attack dataset and contains 115 totals features (5 timestep windows [100 ms, 500 ms, 1.5 s, 10 s, 1 m] and 23 features per timestep window) and over 2.5 million rows of packet data. Multiple histograms and scatter plots were created in the Jupyter Notebook file “Dataset_Visualization – ARP MitM_dataset.ipynb” to display the behavior of malicious versus non-malicious packets with respect to relationships of packet features within respective timesteps. Multiple different plots were crafted to search for possible relationships before any modeling was done. The dataset for ARP MitM Ettercap was separated by its timestep (5 sets of 23 histograms) and the timestep data was graphed as scatter plots and histograms. The plots included with this report exhibit interesting relationships that show that the malicious data is extremely intelligent and blends extremely well with “safe” and “normal” LAN traffic. It is also important to notice that the plots involving more-important features will include somewhat easily separable data like in Figure 1 and Figure 2. These two plots help to show that the spread of some of the data is quite different and that the type of packet can be classifiable. Figure 3 and Figure 4 help to show that the malicious packets tend to follow some of the “safe” packets behavior but fail to follow it exactly. The histograms below show that the data is separable for classification and explain that we could create a model that can utilize these differences in data to classify the malicious packets.

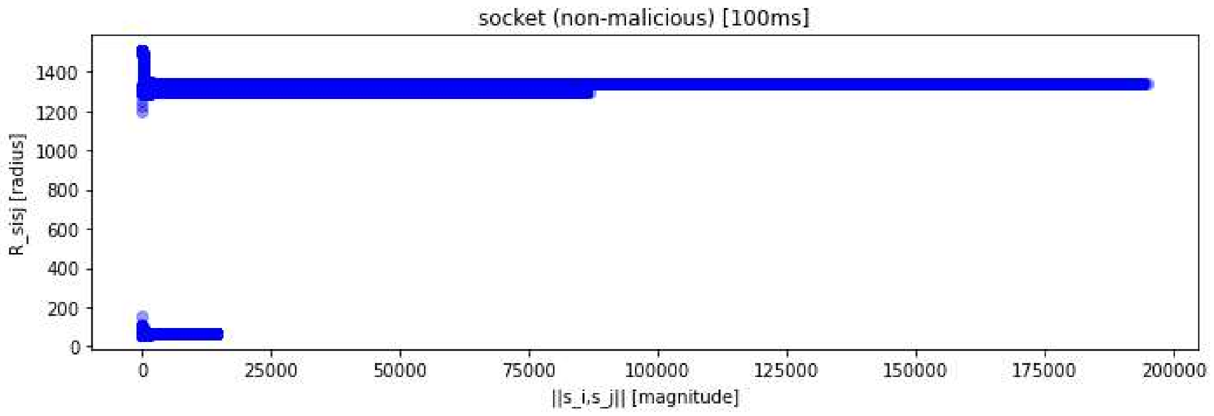


Figure 1. Captured Socket Data for Safe Packet in 100ms Timestep (magnitude vs. radius)

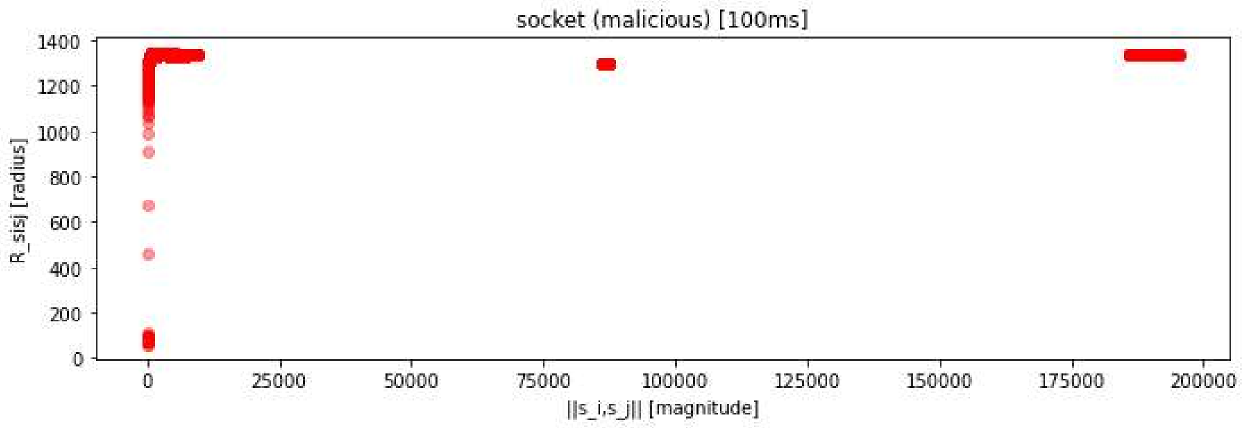


Figure 2. Captured Socket Data for Malicious Packet in 100ms Timestep (magnitude vs. radius)

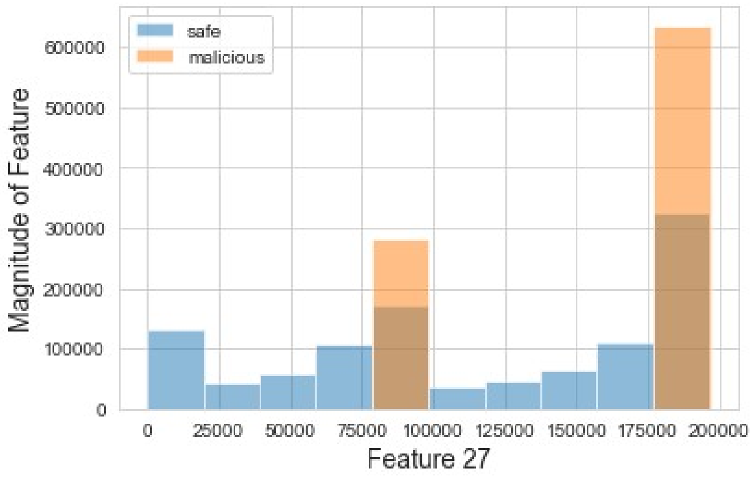


Figure 3. Histogram of the Mean of Channel Source and Destination IP

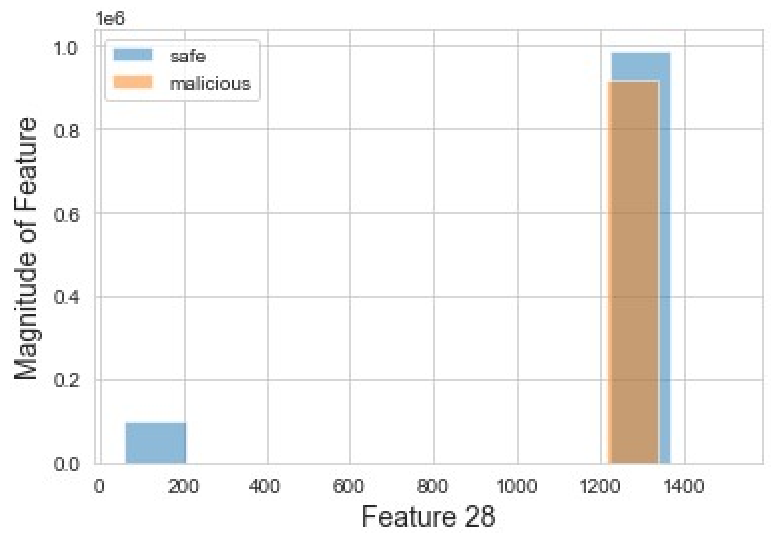
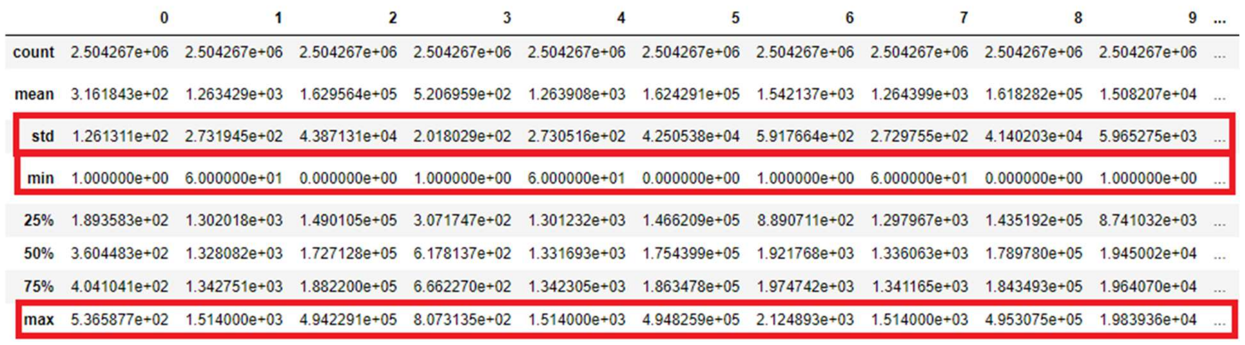


Figure 4. Histogram of the Standard Deviation of Channel Source and Destination IP

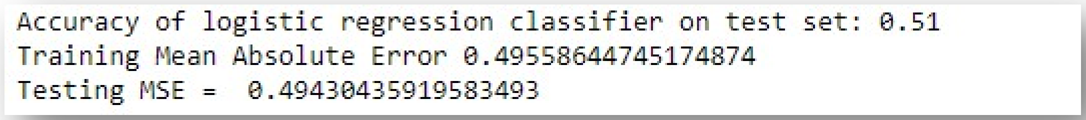
c) Dataset Cleaning

The ARP MitM Ettercap dataset did not have any missing data so no packet information needed to be dropped for this reason. There aren't any serious outliers in the dataset so no dataset cleaning was done for this reason either. Looking at Figure 5, it can be noticed that there aren't any serious problems with standard deviation and the max and min; the quartiles aren't too spread out as well. It should be known that after using Z-score and IQR to clean the dataset, the error rose higher in the final classification results of the model created using the dataset without feature extraction (Figure 6, Figure 7, and Figure 8).



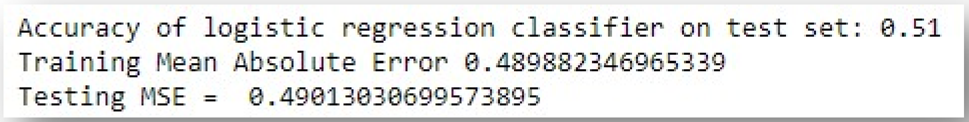
	0	1	2	3	4	5	6	7	8	9	...
count	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	2.504267e+06	...
mean	3.161843e+02	1.263429e+03	1.629564e+05	5.206959e+02	1.263908e+03	1.624291e+05	1.542137e+03	1.264399e+03	1.618282e+05	1.508207e+04	...
std	1.261311e+02	2.731945e+02	4.387131e+04	2.018029e+02	2.730516e+02	4.250538e+04	5.917664e+02	2.729755e+02	4.140203e+04	5.965275e+03	...
min	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	...
25%	1.893583e+02	1.302018e+03	1.490105e+05	3.071747e+02	1.301232e+03	1.466209e+05	8.890711e+02	1.297967e+03	1.435192e+05	8.741032e+03	...
50%	3.604483e+02	1.328082e+03	1.727128e+05	6.178137e+02	1.331693e+03	1.754399e+05	1.921768e+03	1.336063e+03	1.789780e+05	1.945002e+04	...
75%	4.041041e+02	1.342751e+03	1.882200e+05	6.662270e+02	1.342305e+03	1.863478e+05	1.974742e+03	1.341165e+03	1.843493e+05	1.964070e+04	...
max	5.365877e+02	1.514000e+03	4.942291e+05	8.073135e+02	1.514000e+03	4.948259e+05	2.124893e+03	1.514000e+03	4.953075e+05	1.983936e+04	...

Figure 5. Python Dataset Description of ARP MitM Ettercap Dataset



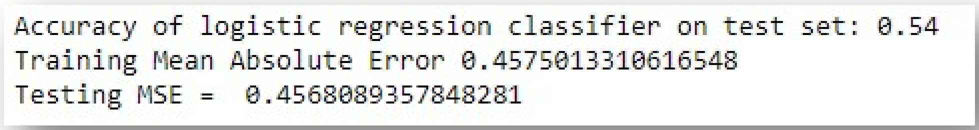
```
Accuracy of logistic regression classifier on test set: 0.51
Training Mean Absolute Error 0.49558644745174874
Testing MSE =  0.49430435919583493
```

Figure 6. Accuracy, Training MSE, and Testing MSE on ARP MitM dataset cleaned with IQR



```
Accuracy of logistic regression classifier on test set: 0.51
Training Mean Absolute Error 0.489882346965339
Testing MSE =  0.49013030699573895
```

Figure 7. Accuracy, Training MSE, and Testing MSE on ARP MitM dataset cleaned with Z-score



```
Accuracy of logistic regression classifier on test set: 0.54
Training Mean Absolute Error 0.4575013310616548
Testing MSE =  0.4568089357848281
```

Figure 8. Accuracy, Training MSE, and Testing MSE on ARP MitM dataset unmodified

d) Related Work

A paper named [1] *Unsupervised Learning Approach for Network Intrusion Detection System Using Autoencoders* written by The Journal of Supercomputing uses the UCI Kitsune Network Attack dataset. This paper aims to solve the problems associated with creating "anomaly detection-based" models that are both supervised and unsupervised. The writers wanted to create models and provide results to give readers some guidelines on how to develop unsupervised malicious packet detection models. They say that acquiring data, like the Kitsune Network Attack dataset, requires professional experience and is typically associated with supervised learning. The researchers involved with the paper wanted to investigate applying unsupervised models to networking classification datasets and show readers some ways to gather data and create and improve unsupervised network detection models. The researchers involved analyzed the nature of the Kitsune Network Attack dataset to show that the model is sound for malicious-packet detection but involves some issues. Their analysis concluded that the dataset consisted of manually-labeled datapoints, “normal” data patterns, and that an unsupervised model would require a manually-set threshold that could be terribly erroneous. The results the researchers obtained state that the Kitsune Network Attack dataset is important for understanding malicious network behavior but it is unrealistic in the sense that it’s too-adjusted and too-clean. They concluded that the dataset is not suited for unsupervised models.

e) Feature Extraction

The Kitsune Network Attack dataset contains an immense 2.5+ million rows of data and 115 columns of features. With 115 features, we had to use feature extraction to increase the model accuracy and decrease the amount of ambiguous data. To better understand the dataset and how it should be modeled to get the best results, we split the data by timestep (every 23 columns in 115 feature set equates to one timestep of data) and found the features that exhibited the highest importance. Doing feature selection was important for increasing accuracy for our classifier because 115 features of data caused a lot of inaccuracy. The main idea of this method was to find the two most important features from each timestep (totaling 10 features for final classification) so that we could focus on the features that truly changed with malicious and safe data. Another possible method would have been to take the same features from every timestep (every timestep has the same feature names) and select the first and second most important timestep found for that feature. Overall, with the current model we were successful in creating an accurate model.

In researching the models, we used two different types of feature extraction to filter out features, including Extra Trees Classifier and Select K Best Classifier. These two methods helped to find the ten most important features in the ARP MitM dataset. The Extra Trees Classifier creates decision trees using training data from the Python input dataframe. Predictions are then made on feature importance by taking the majority vote on the decision tree predictions. The Extra Trees Classifier randomly samples features at every split and fits every decision tree to the whole training dataset. The two most important features for all five timesteps selected by the Extra Trees Classifier can be found in Figure 9. The next feature selection model was the Select K Best Classifier.

The Select K Best Classifier is a univariate feature selection method that selects the most important features based on the “Analysis of Variance Statistical Test” function or “f_classif”. The “Analysis of Variance Statistical Test” or ANOVA under “f_classif” finds the F-statistic or the ratio of variances between many features and then it finds the F-distribution where the program can then find which F-scores are the best. The Select K Best Classifier was used for all 23 features of each of the 5 timesteps and outputted the two features with the highest importance, so we were left with 10 total features to work with in the model generation. The resulting selected features from Select K Best Classifier can be found in Figure 10.

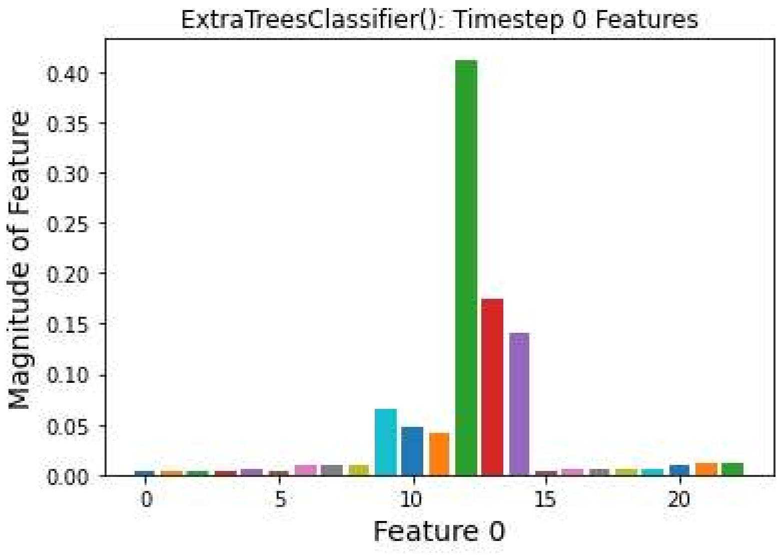


Figure 9. Extra Trees Classifier Selected Feature Results from Training Data

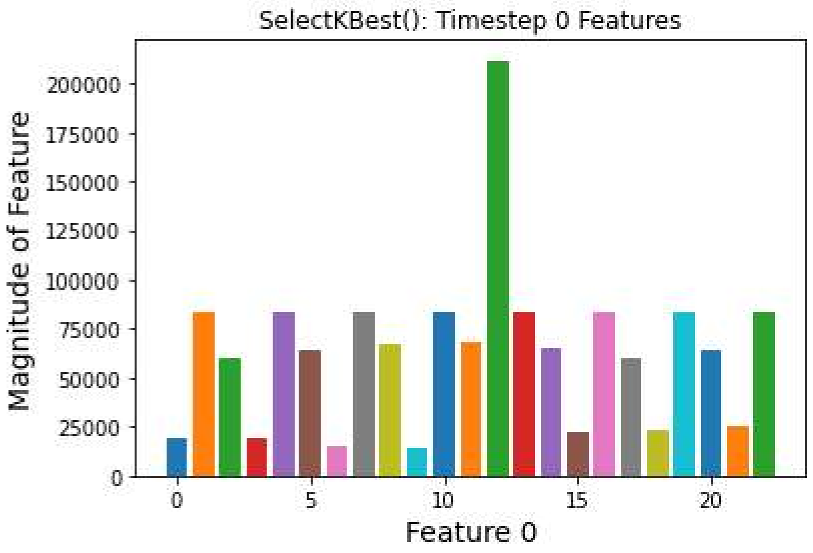


Figure 10. Select K Best Classifier Selected Feature Results from Training Data

f) Classification Model Development

The classification models selected for this project are as follows: Logistic Regression, Random Forest Classifier, LDA, and QDA. These classification models were created in Python using Jupyter Notebooks and utilized input packet data from ARP MitM Dataset which is a part of the Kitsune Network Attack dataset supplied from University of California Irvine. To begin making the model, we loaded the ARP MitM input and pre-classified output ("0" is malicious and "1" is safe) datasets into Jupyter Notebooks. The dataset consists of 115 features and 287,990,705 datapoints total but this is too much data to classify all at once. Before proceeding with model creation, we read the dataset description and found that the 115 features describe the same type of data but in different time frames. The 5 timesteps involved are packets captured in the following windows: 100 milliseconds, 500 milliseconds, 1.5 seconds, 10 seconds, 1 minute into the past and each timestep window contains 23 features:

Outbound Traffic (Label)	Mean (Features)	Standard Deviation (Features)
srcMAC-IP	mew_i	sigma_i
srcIP	mew_i	sigma_i
Channel	mew_i	sigma_i
Socket	mew_i	sigma_i

Table 1. Eight Features of Each Timestep

Outbound and Inbound Traffic (Label)	Magnitude (Features)	Radius (Features)	Covariance (Features)	Correlation Coefficient (Features)
Channel	||s_i,s_j||	R_sisj	cov_sisj	P_sisj
Socket	||s_i,s_j||	R_sisj	cov_sisj	P_sisj

Table 2. Eight Features of Each Timestep

Packet Rate of Outbound Traffic (Label)	Weight (Features)
srcMAC-IP	w_i
srcIP	w_i
Channel	w_i
Socket	w_i

Table 3. Four Features of Each Timestep

Inter-Packet Delays of Outbound Traffic (Label)	Mean (Features)	Standard Deviation (Features)	Weight (Features)
Channel	mew_i	sigma_i	w_i

Table 4. Four Features of Each Timestep

The features involved in Table 1, Table 2, Table 3, and Table 4 make up the 23 features for each of the 5 timesteps; each of the 5 timesteps having 23 features makes up the entire 115 feature dataset. Some extra information about the tables above should be known: information in Table 1 is aggregated into packet size, information in Table 2 is aggregated into packet size, information in Table 3 is aggregated into packet count, and information in Table 3 is aggregated into packet jitter. All the information described can be found in [2] *Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection*.

The dataset on its own is too large, so we decided to divide the data into 5 sets of 23 features for each timestep. Splitting the data in this manner allowed us to see which features were truly important for using in our classification models. Each model was run three times with the following requirements: unmodified data, the feature set modified through Extra Trees Classifier, and the feature set modified through Select K Best Classifier. In the Feature Extraction section of this paper, an alternative method was proposed for feature extraction that could prove to be viable. The proposed method involved comparing the same features from every timestep dataset and finding which 23 features are the most important; this would allow the classification models to utilize every type of feature that we have (even though every feature is only from a single timestep). After splitting each model into 3 separate models, the input data for each method was split into training input and output data and testing input and output data. The training input and output data consisted of 75% of the rows of packet information for training the models, and the testing input and output data consisted of 25% of the rows of packet information for testing the model with “unseen” data. The first model that will be described is the Logistic Regression model.

Logistic Regression is a type of classification method that models the probability that a packet will be malicious or not. Since the output is in a binary classification format, Logistic Regression was an easy model to apply. The equation, found on page 132 in [3] *An Introduction to Statistical Learning with Applications in R*¸ for Multiple Variable Logistic Regression is as follows:

$$p(X) = \frac{e^{\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4+\beta_5X_5+\beta_6X_6+\beta_7X_7+\beta_8X_8+\beta_9X_9+\beta_{10}X_{10}}}{1+e^{\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\beta_4X_4+\beta_5X_5+\beta_6X_6+\beta_7X_7+\beta_8X_8+\beta_9X_9+\beta_{10}X_{10}}}$$

Equation 1. Multiple Variable Logistic Function Equation with 10 Features

From the equation, β are the feature coefficients and X consists of the input data associated with the correctly numbered feature. For this paper, the Logistic Regression model was created three times for modeling: once for no data modification (115 features), once for data selected through Extra Trees Classifier (10 features), and once for data selected through Select K Best Classifier (10 features). Before creating the model, the input dataset were cut into a 75-25 split using "train_test_split()" Python function which can throw data into bins either randomly or pseudo-randomness based on a see. The resulting training data consisted of 75% of the input data scrambled with respect to a known seed, and the resulting testing data consisted of 25% of the input data also scrambled with respect to a known seed. The "sklearn" Python package makes using this model very easy; to create the model, all that needs to be done is a variable is set equal to the "LogisticRegression()" model function and then you fit the data to it. The Logistic Regression model used the following features: multivariate via "multi_class", the 'lbfgs' solver, L2 penalty, and regularization strength of 1.0. At this point, the model is now trained and the training MSE's were found to be: 45.17% for the unmodified data, 7.39% for feature selection with Extra Trees Classifier, and 1.02% for feature selection with Select K Best Classifier. The next model that will be discussed in this paper is the Random Forest Classifier.

Random Forest Classification is a type of classification algorithm that uses randomized binary decision trees to create a "forest" of tree possibilities. The trees are trained with the bagging method which using general bootstrapping. Random Forest was used in this project because the high amount of randomness created through the classifier allows a model to see many possibilities of different outcomes and create a model that's good at predicting situations. With a lot of features and a lot of datapoints, Random Forest Classifier should provide good results due to its ability to create high amounts of randomness. The randomness for this dataset is important because of its high amount of datapoints. For this paper, the Random Forest Classifier model was run three times for modeling: one for no data modification, once for data selected through Extra Trees Classifier, and once for data selected through Select K Best Classifier. The data fed into the models were split 75% as training data and 25% as testing data. Next, using the Python "sklearn" package we created a Random Forest Classifier model with "RandomForestClassifier()" function by setting it equal to a variable. Using this variable that contains the model function, we fit the model using the training data. At this point the model is now trained and the training MSE was found to be 0% for all three Random Forest Classifier models we created. This could be concerning but we won't know until testing data is predicted using the model. The next model that will be discussed in this paper is the LDA classifier.

The LDA Classifier stands for Linear Discriminant Analysis and it's a type of classifier that uses feature statistics to try and find a linear combination between highly separated and normalized data. This is an extension/more well-defined linear classifier than Logistic Regression, so we used it to compare the results to Logistic Regression to note whether or not the data can be summarized in a linear fashion. The model uses a multivariate and class-specific

Gaussian distribution of data instead of the unformatted data used in Logistic Regression and makes parameter assumptions by using Bayes' Theorem. The LDA model should provide results that are a little better than what Logistic Regression but the expectation is that the model doesn't operate best with a linear model. For this paper, the LDA Classifier model was run three times for modeling: one for no data modification, once for data selected through Extra Trees Classifier, and once for data selected through Select K Best Classifier. The data fed into the models were split 75% as training data and 25% as testing data. Next, using the Python "sklearn" package we created a LDA Classifier model with "LinearDiscriminantAnalysis()" function by setting it equal to a variable .The Python LDA model used the 'svd' solver and all other settings were default. Using this variable that contains the model function, we fit the model using the training data. At this point, the model is now trained and the training MSE's were found to be: 7.64% for the unmodified data, 10.84% for feature selection with Extra Trees Classifier, and 10.84% for feature selection with Select K Best Classifier. The last model that is described in this paper is the QDA classifier.

The QDA classifier stands for Quadratic Discriminant Analysis and it's a classifier that utilizes quadratic-type functions instead of linear. QDA is very similar to LDA because it uses a Gaussian distribution of datapoints for each output class and also uses the Bayes' Theorem to create feature assumptions. QDA uses a covariance matrix for each feature, so even though it may be more computationally expensive it should give accurate results that will use the relationships between each feature that is finds. Like the Random Forest Classifier, the decision boundary is nonlinear and will give us more insight on whether nature of this dataset is linear or nonlinear. The QDA classifier model was run three separate times for modeling: one for no data modification, once for data selected through Extra Trees Classifier, and once for data selected through Select K Best Classifier. Before creating the model, the data was randomly split using a 75% training data and 25% testing data split in Python. Next, the model was trained using "fit()" in Python and the training MSE's were: 2.62% for the unmodified data, 14.90% for feature selection with Extra Trees Classifier, and 1.06% for feature selection with Select K Best Classifier. Next, the models were evaluated using testing data and model optimizations were applied.

g) Fine-Tune your models & Feature Set:

Now that the models had been trained, they needed to be evaluated with testing data. Each of the 4 models: Logistic Regression, Random Forest Classifier, LDA, and QDA had been tested with the data they were trained with and the results were verified. After that, the 3 models of each model were tested with the dataset testing data or "unseen" data to verify that the model has some accuracy with new data. The first model discussed is the Logistic Regression model.

The Logistic Regression models were fit with the training data and then tested again with the training data. The results were as follows:

- Before Optimization:
    - No Feature Selection
        - Performance: Terrible
            - 54% accuracy on the training data
            - 54% accuracy on the testing data
    - Feature Selection using Extra Trees Classifier
        - Performance: Good
        - 92.6% accuracy with training data
        - 92.61% accuracy with testing data
    - Feature Selection using Select K Best Classifier
        - Performance: Decent
        - 99% accuracy with training data
        - 99% accuracy with testing data

Some of the training accuracies and the testing accuracy on the unmodified data were very bad. Some of this most likely had to do with how the data was split into training and testing data. Explanations and reasoning behind how things changed with the applied optimizations will be discussed in section “i)”. The following information involves each model, how it was optimized, and what its results were:

- After Optimization:
    - No Feature Selection
        - Applied Optimization: None
        - Reasoning: 115 features is too much for Logistic Regression
        - We tried to change inverse regularization strength, we tried to balance the training set, tried a different classifier solver, and also tried a different train-test split
    - Feature Selection using Extra Trees Classifier
        - Applied Optimization: Increase in C, the inverse of the regularization strength, change the train+test split to a different randomization seed, feature optimization through Extra Trees Classifier
        - 94.47% accuracy with training data
        - 94.49% accuracy with testing data
    - Feature Selection using Select K Best Classifier
        - Applied Optimization: None
        - Reasoning: model already had great performance
        - 99% accuracy with training data
        - 99% accuracy with testing data

The Random Forest Classifier models were fit with the training data and then tested again with the training data. After the results from predicting the training data, we used the model to predict the testing or "unseen" data. The results were as follows:

- Results:
  - No Feature Selection
    - Performance: Great
    - 100% accuracy on the training data
    - 99.99% accuracy on the testing data
  - Feature Selection using Extra Trees Classifier
    - Performance: Good
    - 100% accuracy on the training data
    - 99.99% accuracy on the testing data
  - Feature Selection using Select K Best Classifier
    - Performance: Decent
    - 100% accuracy with training data
    - 99.99% accuracy with testing data

The results were extremely accurate and could possibly be pushed even further but modifying the models any further was unnecessary for this project. The feature optimization did not seem to provide any help to the Random Forest Classifier.

The Linear Discriminant Analysis or LDA Classifier were fit with training data and then tested again with the training data. After fitting the model and using it to predict the training data, we used the model to try to predict the testing data. The results were as follows:

- Results:
  - No Feature Selection
    - Performance: Good
    - 92.35% accuracy on the training data
    - 92.27% accuracy on the testing data
  - Feature Selection using Extra Trees Classifier
    - Performance: Good
    - 89.16% accuracy on the training data
    - 89.18% accuracy on the testing data
  - Feature Selection using Select K Best Classifier
    - Performance: Good
    - 89.16% accuracy with training data
    - 89.18% accuracy with testing data

The LDA Classifier accuracy was fairly high for every model, but the feature selection seemed to hurt the training and testing accuracy overall. Since the feature selections did not provide any advantage, we chose not to optimize them. The following information describes the model with no feature selection, how it was optimized, and what its results were:

- After Optimization:
  - No Feature Selection
    - Applied Optimization: Applied Z Score and only allowed rows that contained values with Z Scores under 2.
    - 99.98% accuracy on the training data
    - 99.98% accuracy on the testing data
  - Feature Selection using Extra Trees Classifier
    - Applied Optimization: None
    - Reasoning: Worse performance than unmodified input data
  - Feature Selection using Select K Best Classifier
    - Applied Optimization: None
    - Reasoning: Worse performance than unmodified input data

The Quadratic Discriminant Analysis or LDA Classifier were fit with training data and then tested again with the training data. After fitting the model and using it to predict the training data, we used the model to try to predict the testing data. The results were as follows:

- Results:
    - No Feature Selection
        - Performance: Great
        - 97.38% accuracy on the training data
        - 96.43% accuracy on the testing data
    - Feature Selection using Extra Trees Classifier
        - Performance: Good
        - 85.10% accuracy on the training data
        - 72.03% accuracy on the testing data
    - Feature Selection using Select K Best Classifier
        - Performance: Decent
        - 98.94% accuracy with training data
        - 89.82% accuracy with testing data

The QDA models all had high training and testing accuracy. The two models involving no feature selection and Select K Best Classifier feature selection had the highest amount of prediction accuracy, so we continued by trying to optimize those models. We decided to not optimize the model that included the Extra Trees Classifier because it had worse performance and even gave an error stating that the input data was "collinear" for the QDA function. The following information describes the models with no feature selection and Select K Best Classifier, how they were optimized, and what the overall results were:

- Results:
    - No Feature Selection
        - Applied Optimization: Applied Z Score and only allowed rows that contained values with Z Scores under 2.
        - 99.84% accuracy on the training data
        - 99.91% accuracy on the testing data
    - Feature Selection using Extra Trees Classifier
        - Applied Optimization: None
        - Reasoning: Worse performance than unmodified input data, data inputs considered "collinear" which is bad for QDA
    - Feature Selection using Select K Best Classifier
        - Applied Optimization: None
        - Reasoning: When trying to apply parameter adjustment, the "variables are collinear" warning came up and the training and testing accuracy became poor.

h) Performance

- <u>The Logistic Regression Model Performance Results:</u>
    - Model using Select K Best Classifier Feature Selection (Highest Accuracy)
        - Training MSE: 0.944% (classification of 75% of random data)
        - Test MSE: 0.9296% (classification of 25% of random "unseen" data)
        - Resulting Test Confusion Matrix:

	Predicted Negative	Predicted Positive
True Negative	672,050	7,448
True Positive	4,192	568,444

Table 5. Confusion Matrix for Logistic Regression Model

- ROC Curve:

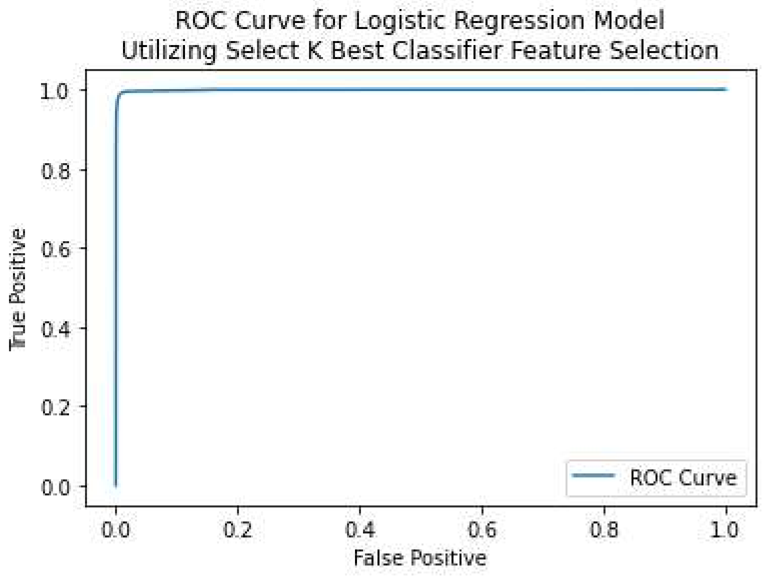


Figure 11. ROC Curve for Logistic Regression Model

- Classification Report and Accuracy Score Tables:

Training Data	Precision	Recall	F1-Score	Support	Result
Not malicious	0.99	0.99	0.99	679497	
Malicious	0.99	0.99	0.99	572636	
Accuracy			0.99	1252133	0.9905577
Macro AVG	0.99	0.99	0.99	1252133	
Weighted AVG	0.99	0.99	0.99	1252133	

Table 6. Classification and Accuracy Reports for Logistic Regression Training Data

Test Data	Precision	Recall	F1-Score	Support	Result
Not malicious	0.99	0.99	0.99	679498	
Malicious	0.99	0.99	0.99	572636	
Accuracy			0.99	1252134	0.99070387
Macro AVG	0.99	0.99	0.99	1252134	
Weighted AVG	0.99	0.99	0.99	1252134	

Table 7. Classification and Accuracy Reports for Logistic Regression Test Data

- <u>The Random Forest Classifier Model Performance Results:</u>
  - Model using no Feature Selection (Highest Accuracy)
    - Training MSE: 0.0% (classification of 75% of random data)
    - Test MSE: 7.986e-04% (classification of 25% of random "unseen" data)
    - Resulting Test Confusion Matrix:

	Predicted Negative	Predicted Positive
True Negative	340,204	2
True Positive	3	285,858

Table 8. Confusion Matrix for Random Forest Classifier Model

    - ROC Curve:

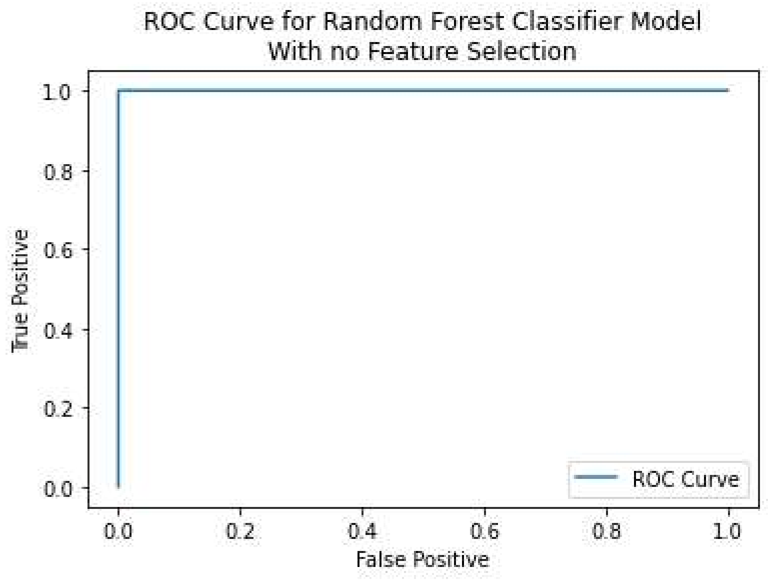


Figure 12. ROC Curve for Random Forest Classifier

- Classification Report and Accuracy Score Tables:

Training Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	1018789	1.0
Malicious	1	1	1	859411	
Accuracy			1	1878200	
Macro AVG	1	1	1	1878200	
Weighted AVG	1	1	1	1878200	

Table 9. Classification and Accuracy Reports for Random Forest Training Data

Test Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	340206	0.99999
Malicious	1	1	1	285861	
Accuracy			1	626067	
Macro AVG	1	1	1	626067	
Weighted AVG	1	1	1	626067	

Table 10. Classification and Accuracy Reports for Random Forest Test Data

- The LDA Classifier Model Performance Results:
    - Model using no Feature Selection (Highest Accuracy)
        - Training MSE: 0.01795% (classification of 75% of random data)
        - Test MSE: 0.015988% (classification of 25% of random "unseen" data)
        - Resulting Test Confusion Matrix:

	Predicted Negative	Predicted Positive
True Negative	502241	6
True Positive	157	567135

Table 11. Confusion Matrix for LDA Classifier Model

- ROC Curve:

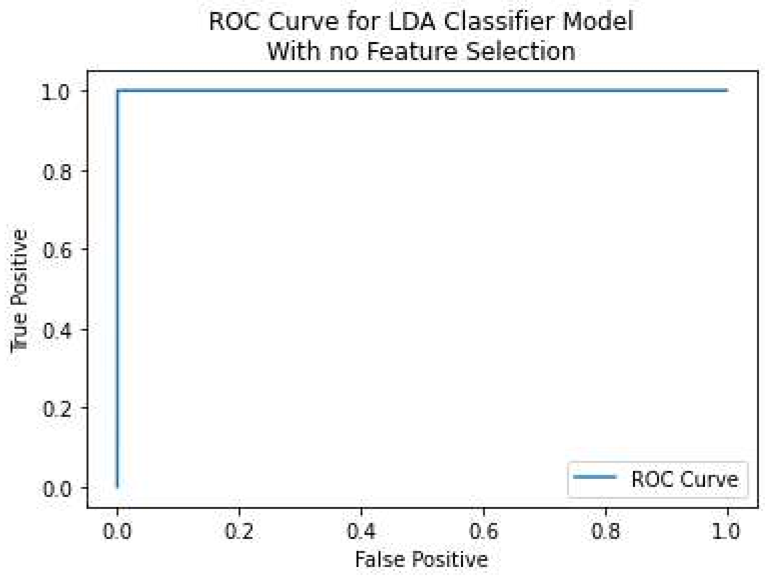


Figure 13. ROC Curve for LDA Classifier

- Classification Report and Accuracy Score Tables:

Training Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	167066	
Malicious	1	1	1	189446	
Accuracy			1	356512	0.99982
Macro AVG	1	1	1	356512	
Weighted AVG	1	1	1	356512	

Table 12. Classification and Accuracy Reports for LDA Training Data

Test Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	502247	0.99984
Malicious	1	1	1	567292	
Accuracy			1	1069539	
Macro AVG	1	1	1	1069539	
Weighted AVG	1	1	1	1069539	

Table 13. Classification and Accuracy Reports for LDA Test Data

- <u>The QDA Classifier Model Performance Results:</u>
    - Model using no Feature Selection (Highest Accuracy)
        - Training MSE: 0.161% (classification of 75% of random data)
        - Test MSE: 0.0897% (classification of 25% of random “unseen” data)
        - Resulting Test Confusion Matrix:

	Predicted Negative	Predicted Positive
True Negative	100215	0
True Positive	358	113335

Table 14. Confusion Matrix for QDA Classifier Model

- ROC Curve:

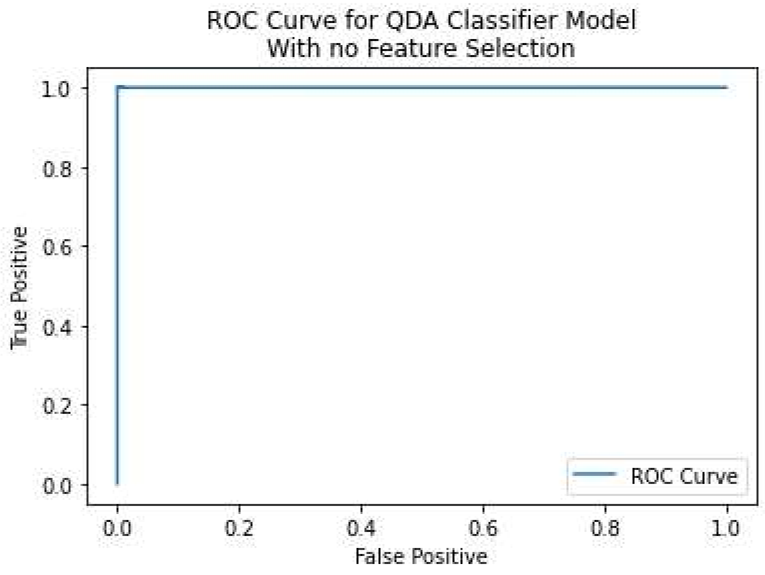


Figure 13. ROC Curve for QDA Classifier

- Classification Report and Accuracy Score Tables:

Training Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	569098	0.998389
Malicious	1	1	1	643045	
Accuracy			1	1212143	
Macro AVG	1	1	1	1212143	
Weighted AVG	1	1	1	1212143	

Table 12. Classification and Accuracy Reports for QDA Training Data

Test Data	Precision	Recall	F1-Score	Support	Result
Not malicious	1	1	1	100215	0.999102
Malicious	1	1	1	113693	
Accuracy			1	213908	
Macro AVG	1	1	1	213908	
Weighted AVG	1	1	1	213908	

Table 13. Classification and Accuracy Reports for QDA Test Data

i) Overall Discussion of Results and Conclusions

In total, the results for each model were not very good to start with before optimization. The dataset has too many datapoints and too many features for a type of model to be able it fit without modification. After optimization applied on both the dataset and the models, each model reached around 99% accuracy. The Logistic Regression model performed very well after multiple different optimizations.

The Logistic Regression model rose in optimization after modifying the data and the model itself. The reason why the Logistic Regression model did very poorly with no dataset modification and no model modification is because the dataset was just too large for the typical model to converge accurately. To overcome this problem, a lot of the dataset was removed by finding the 10 most important features and only using those 10 columns of data in classification. Selecting features with Select K Best Classifier helped the Logistic Regression model because it finds the linear relationship between features and the result and gives an importance score based on the magnitude of the relationship. Select K Best works so well with Logistic Regression because Logistic Regression is considered a linear model. The results from Select K Best classifier greatly improved the results of the Logistic Regression model and therefore no other optimizations for this model were done, but the model utilizing Extra Trees Classifier did however increase in optimization after increasing the inverse regularization strength. Changing the regularization strength helped to loosen the prediction fit and therefore gave a higher accuracy score for both training and testing data. The next model discussed is the Random Forest Classifier.

The Random Forest Classifier was optimized the first time running it because it is an extremely diverse algorithm. The classifier was not given a depth restriction so it could run until complete saturation. This model received over 99.99% accuracy on both training and testing data on all three models that we created. This model is computationally expensive, but it did compute faster than logistic regression. The versatility and randomness of the model makes this a very good method of evaluating the Kitsune Network Attack dataset. The next model discussed is the LDA model.

The LDA classifier had high accuracy for models including unmodified data, feature selection with Extra Trees Classifier, and feature selection with Select K Best Classifier. Looking at the accuracy results from unmodified data to the models using feature selection, it seems as though feature selection hurt the accuracy of the model. Just like Logistic Regression, the LDA model is linear and tends to search for the linear relationship between features and the output. Before applying any optimization, the model with no feature selection had an accuracy score of 92.35% on training data and 92.27% on testing data. This accuracy is fairly good even with the amount of data in dataset and with the number of features. To get the accuracy up to 99.98%, we applied the Z-score technique to only take data that was within 2 standard deviations of the mean of a column of data. Taking the Z-score of input data rose the model performance from

92.27% up to 99.98%; the Z-score helped the overall model because it provided less variance in the input data which causes linear inaccuracies. The final model discussed is the QDA model.

The QDA classifier performed well before and after applying optimizations. The QDA models utilizing no feature selection and Select K Best feature selection performed very well before applying further optimizations. After applying the Z-score to the model with no feature selection, the model rose up to 99% accuracy for training and testing data. The Z-score modified the input data by only allowing rows of data that contained data that fulfilled the requirement of having data that is less than 2 standard deviations from the overall mean of column data. This model performed very well because it should perform well on models that LDA does well on. This model is meant to fit and classify nonlinear data and possibly linear data.

Overall, every model did well in classifying the Kitsune Network Attack dataset. The dataset seems to have a linear nature to it, hence why Logistic Regression and LDA had high performances. Out of the 4 models described in this paper, choosing a model to use (after the fact) would involve a model that is less computationally expensive and that can be applied with minimal interference. The Logistic Regression model should not be used because there is too much data and too many features to allow it to be used without modification. The Random Forest is computationally expensive but it’s the most accurate classifier and works out-of-the-box with no modification. The LDA model has a high accuracy and performs well with minimal optimization; the LDA model is expensive but did run fairly quickly for this assignment. The QDA model should not be used because it was found that the dataset could be quantified linearly. In conclusion, the Random Forest and LDA models should be used for this dataset.

References

[1] Choi, H., Kim, M., Lee, G. *et al.* Unsupervised learning approach for network intrusion detection system using autoencoders. *J Supercomput* **75,** 5597–5621 (2019). https://doi.org/10.1007/s11227-019-02805-w

[2] Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: an ensemble of autoencoders for online network intrusion detection.

[3] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai, 'Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection', Network and Distributed System Security Symposium 2018 (NDSS'18)