PROJECT DELIVERABLE II REPORT
Cyber Attack Detection - Project 4

UG Team 6
Aliya Flanigan, Morgan Harrison, Linda Valencia, Michelle Yin

Project Description

Data analytics plays an important role in Cyber Security. Data is collected from computers, networks, cloud systems, etc., which needs to be evaluated when attempting to identify impending attacks. With the assistance of data analytics, early detection of this is possible and provides a chance to subside threats before illegal breaches have occurred or can occur. The data that was provided displayed numerical features that were extracted from network flows and used to classify each network activity as either a benign activity or a type of attack, with a total of 8 distinct attack types and 1 benign activity type. There were a total of 760,000 samples and 78 predictors. It was necessary to remove 10 of the 78 predictor variables due to the values being NAN or zero. This is further described in the data visualization and data pre-processing sections. The objective of this project is to appropriately predict the type(s) of attack(s) based on a series of relevant predictor and response variables. In order to successfully accomplish this objective, one must initially visualize the data to obtain a more comprehensive understanding of the raw information and data pre-processing and splitting techniques should be implemented. Following this, it was appropriate to implement and execute multinomial logistic regression to determine the variables that may or may not be threatening to the overall system and assist in the classification of said variables. The results of the model and its analysis need to be visually presented for complete understanding and further use in assistance in preventing possible attacks.

Data Visualization

Following the project interpretation and preceding the data pre-processing and splitting techniques; it was necessary to gain a better understanding of the raw data that was first collected and then analyzed. This was first initialized by establishing a bar chart (*Figure 1*) to comprehend and evaluate the flexibility of the data and also gain a visualization across the categories. Following the bar chart, a pie chart (*Figure 2*) was set up to determine the composition of the data as a whole and more easily identify the different predictor variables (proportionally) compared to other predictor variables. From this, it was easy to determine that most of the significant predictor variables were benign, but further visualization was still necessary. In order to obtain a more comprehensive understanding of the data set and its correlational interactions, a correlation matrix (*Figure 3*) was applied. After initial analysis, it was more appropriate to first remove all columns with "NAN" and zero values to better correlate the data. This was then followed by a second correlation matrix (*Figure 4*) that helped identify predictor variables that were highly correlated (positive linear relationship), thus implying that when certain variables interact with one another they result in consistent specific outcomes. Following the implementation of two correlation matrices, a heat map (*Figure 5*) was utilized to represent the frequency of all the different predictor variables in two dimensions. This provided an easier illustration of the relationship interactions in conjunction with the correlation matrices previously established. Lastly, a pair plot (*Figure 6*) was established to further understand these correlations by allowing the interpretation of trends and patterns within the correlated data set. Additionally, it provided some insight for outliers within the dataset that may need to be normalized, further scaled, or dropped in the model.

Figure 1- Bar Chart

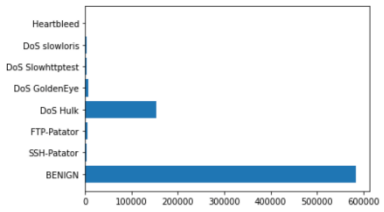


Figure 2- Pie Chart

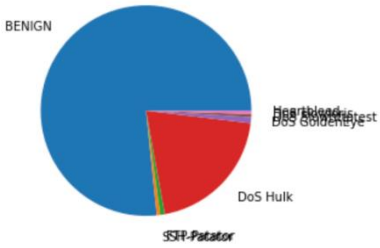


Figure 3- Correlation Matrix

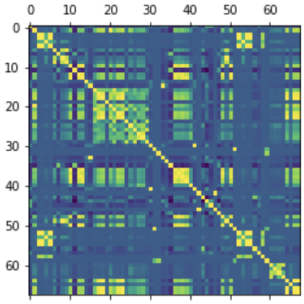


Figure 4- Correlation Matrix (Highly Correlated Pairs)

Figure 5- Heat Map

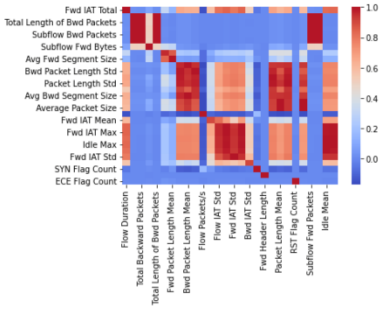
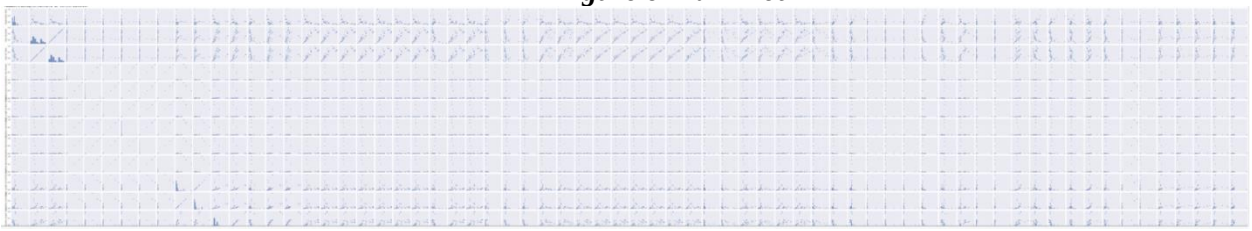


Figure 6- Pair Plot



Data Pre-Processing

To make sure the raw data was suitable for analysis, it was necessary to perform feature extraction and scale the data. In order to perform feature extraction, it was necessary to remove the "NAN" and zero values from the data, in order to create a structured dataset that would be suitable for statistical analysis. If a predictor indicates that 700,000 plus rows are zero, it is unlikely to have a significant impact on the data. Furthermore, min-max scaling was used to scale the data, which involved bringing the values of variables into a common range and scale of zero to one. This method helped preserve the shape of the distribution and maintain the relative distance between values. By implementing this approach, the analysis was able to keep important information, such as the presence of outliers, skewness, and the distance between values.

Data Splitting

In this study, a test set size of 0.20 was chosen for data splitting using the train_test_split function in order to allocate 20% of the data for model evaluation and to ensure a sufficient amount of training data to avoid overfitting. However, the model was executed using various testing sizes to verify the accuracy of the model. On the following page, there is a visual representation of the results for the test size and accuracy of the data splitting using logistic regression:

Test Size	Accuracy
15%	0.9546102890345389
20%	0.9537569978145161
25%	0.9549986611432262

Logistic Regression

Model Selection

As the training data set is classified as a classification model, it could be associated as a binary logistic regression, multinomial logistic regression, or ordinal logistic regression. As the target variable has more than three nominal categories, the best fit is multinomial logistic regression. To account for the large dataset, the penalty on L2 and solver on SAG were specified. This penalty provides a smoother and more stable solution for large numbers of training samples. Moreover, L2 handles correlated predictors better than L1 which could lead the model to have a more accurate prediction. SAG is useful for large data sets, as it maintains a running average of past gradient values and uses them to update the model parameters, providing faster convergence and better generalization performance than other solvers. Furthermore, it was defined as the maximum number of iterations by creating a balance between the performance of the model and the computational time.

Resampling Method Utilized During Training

After validating the model through a data splitting of 20%, the model resulted in an accuracy of 96%. Then, stratified K-fold cross-validation was performed, as it was noticed that a larger percentage of the target values belonged to “BENIGN” (76.6%) and “DoS Hulk” (20.2%). Given that six classes accounted for 3.8% of the target values for all of the samples, the goal was to preserve the proportion of each class in the original dataset. Therefore, after executing the stratified K-fold cross-validation, the following scores were obtained:

```
Cross Validation Scores are [0.94350594 0.94198333 0.94645857 0.93984301
0.94329518 0.96255119 0.96232805 0.96161924 0.96133046 0.96190801]
Average Cross Validation score :0.952482298790731
```

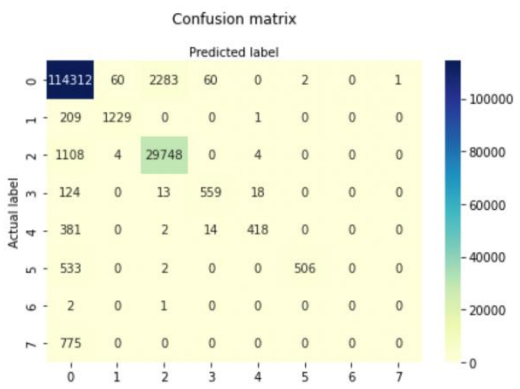
Performance Metric for Prediction Accuracy of the Model

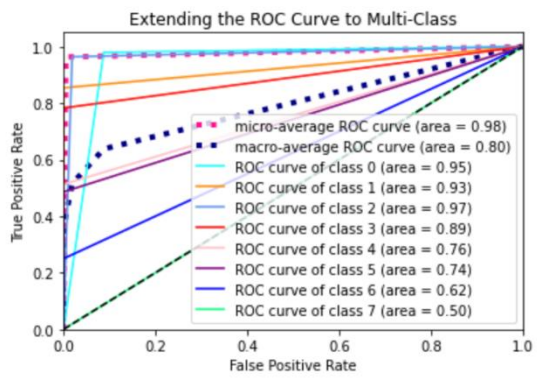
Accuracy was chosen as one of the performance metrics for this project to evaluate the prediction accuracy of the classification model. It measures the proportion of correctly classified instances out of the total number of instances, providing an intuitive and easily interpretable measure of the model's performance. The accuracy obtained for the data without scaling was 87%, when performing standard scaler the accuracy resulted in 95%, and then executing min-max scaler the final accuracy resulted in 96%.

The report presents the results of precision and recall. The precision value obtained was 0.96 using average=micro and 0.70 using average=macro, whereas the recall value obtained was 0.96 using average=micro and 0.57 using average=macro. Micro averaging calculates the metrics globally by counting the total true positives, false negatives, and false positives and then computing precision and recall from these counts. This method might be useful when having imbalanced datasets (one class is more dominant than others), which is the case for this dataset. On the other hand, macro averaging calculates the metrics for each class separately and then averages them, giving equal weight to each class. This method is useful when having a balanced evaluation of the model's performance across all classes.

Visual Representation of Results

A. Confusion Matrix	***B. ROC Curve***
A confusion matrix was made to cross-tabulate the observed and predicted classes for the data.	The following ROC represents the performance of a multiclass classifier system as its discrimination threshold is varied.





C. Metrics

After performing Stratified Cross-Validation, the average accuracy resulted in 0.95, which is considered to be a high performance value. On the other hand, by splitting the data using a testing size of 25% the accuracy resulted in 0.95, while scaling the data using standard scaler. However, when using min-max scaler, the accuracy improved to 0.96.

	precision	recall	f1-score	support
BENIGN	0.97	0.98	0.98	116718
DoS GoldenEye	0.95	0.85	0.90	1439
DoS Hulk	0.93	0.96	0.95	30864
DoS Slowhttptest	0.88	0.78	0.83	714
DoS slowloris	0.95	0.51	0.67	815
FTP-Patator	1.00	0.49	0.65	1041
Heartbleed	0.00	0.00	0.00	3
SSH-Patator	0.00	0.00	0.00	775
accuracy			0.96	152369
macro avg	0.71	0.57	0.62	152369
weighted avg	0.96	0.96	0.96	152369

Random Forest

Model Selection

Random Forest was chosen as it is a robust and scalable algorithm that can handle large and complex datasets, such as this one. Individual models are trained to solve the same problem and then aggregated to make a final prediction. Thus, if one tree happens to miss the minority class samples, another tree in the forest may still capture them, leading to a more accurate prediction overall. By using HalvingGridSearch, the hyperparameters utilized were {'criterion': 'entropy', 'n_estimators': 100}.

Resampling Method Utilized During Training

Random Forest utilizes bagging as a resampling method, thus aggregating the predictions from all the models to make a final prediction. Additionally, HalvingGridSearch used cross-validation, assessing its performance on multiple variations of the data.

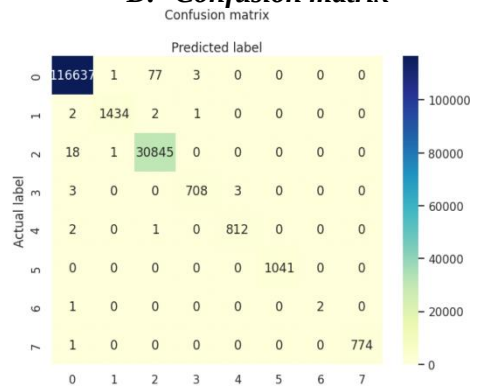
Performance Metric for Prediction Accuracy of the Model

```
Accuracy: 0.9992386902847692
Precision (micro): 0.9992386902847692
Precision (macro): 0.9983112431944738
Recall (micro): 0.9992386902847692
Recall (macro): 0.9992386902847692
```

Previously, it was noticed that the model needed to better identify minority classes. For this deliverable, the goal was to increase the Recall, which measures the proportion of actual positive samples that are correctly identified by the model.

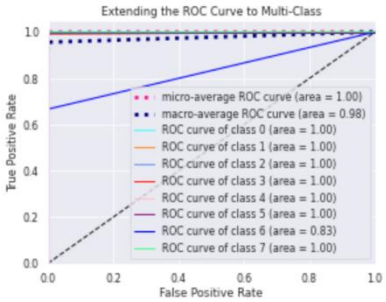
Visual Representation of Results

D. Confusion matrix



E. ROC curve

All of the ROC curves, except for class 6, have an area of 1, meaning that the model can almost perfectly distinguish between the classes.



F. Metrics

	precision	recall	f1-score	support
BENIGN	1.00	1.00	1.00	116718
DoS GoldenEye	1.00	1.00	1.00	1439
DoS Hulk	1.00	1.00	1.00	30864
DoS Slowhttptest	1.00	0.99	0.99	714
DoS slowloris	1.00	1.00	1.00	815
FTP-Patator	1.00	1.00	1.00	1041
Heartbleed	1.00	0.67	0.80	3
SSH-Patator	1.00	1.00	1.00	775
accuracy			1.00	152369
macro avg	1.00	0.96	0.97	152369
weighted avg	1.00	1.00	1.00	152369

Support Vector Classifier

Model Selection

Based on the characteristics of the data set, a Support Vector Classifier (SVC) was also tested as the model for classification. SVMs can capture complex non-linear relationships in the data, which makes it an appropriate choice for this particular data set. Additionally, 20% of the data was used to train the model using a HalvingGridSearchCV to find the best hyperparameter for C between 1, 10 and 100, which resulted in {'C': 100, 'kernel': 'rbf'}.

Resampling Method Utilized During Training

During HalvingGridSearch, cross-validation was set to two folds. Splitting the dataset improves the ability of the model to generalize well on new data that it has not seen before.

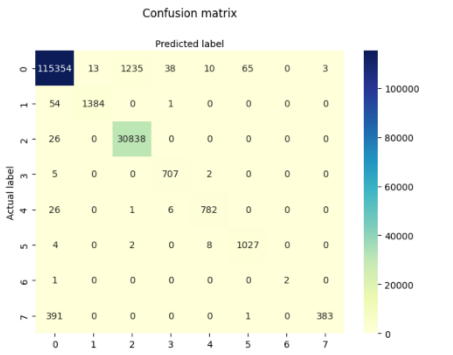
Performance Metric for Prediction Accuracy of the Model

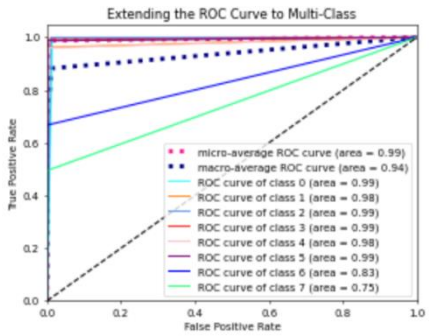
```
Accuracy: 0.9875827760239944
Precision (micro): 0.9875827760239944
Precision (macro): 0.9743485264112457
Recall (micro): 0.9875827760239944
Recall (macro): 0.9875827760239944
```

SVC is able to handle complex datasets, however, it can undergo performance issues when working with very large data sets. So, although the prediction accuracy is respectable, it is not as desirable as the one generated by the random forest model.

Visual Representation of Results

G. Confusion matrix	*H. ROC curve*





I. Metrics

	precision	recall	f1-score	support
BENIGN	0.99	0.98	0.99	467131
DoS GoldenEye	0.98	0.96	0.97	5517
DoS Hulk	0.94	1.00	0.97	123352
DoS Slowhttptest	0.93	0.97	0.95	2947
DoS slowloris	0.97	0.86	0.91	3106
FTP-Patator	0.94	0.99	0.96	4254
Heartbleed	1.00	0.67	0.80	6
SSH-Patator	1.00	0.49	0.66	3161
accuracy			0.98	609474
macro avg	0.97	0.87	0.90	609474
weighted avg	0.98	0.98	0.98	609474

Conclusion

In conclusion, the use of a min-max scaler proved to be helpful in increasing the accuracy of the model. The scaling technique allowed the model to learn the patterns in the data more effectively by bringing all the features to the same scale. As a result, the model was able to make better predictions, leading to an increase in accuracy. The results from the logistic regression model revealed that the precision and recall values with micro averaging are equivalent, indicating a satisfactory model performance across all classes. However, using macro averaging demonstrates lower precision and recall for some classes. Additionally, it was recognized that the model was unable to predict any value for the class ‘Heartbleed’, implying that the dimensions of the dummy variables needed for the augmentation of the ROC curve were not consistent. Therefore, further enhancement in the model's performance on those classes was necessary and appropriate for attaining a more advanced model.

Consequently, two effective classification algorithms, random forest and support vector classifier, were implemented to increase recall in a classification problem, with the reason for choosing recall under the performance metrics section for random forest. Random Forest can be effective in increasing recall because it tends to perform well on imbalanced datasets, as each decision tree is built on a random subset of features and data points, which helps to reduce overfitting and increase the accuracy of the model. SVC is a type of linear classification algorithm that works by finding the hyperplane that best separates the different classes in the data, such that the separating hyperplane is the farthest distance from the training observations. It can be effective in increasing recall because it is capable of handling datasets that are difficult to separate by transforming the data into a higher-dimensional space. However, while executing the code for SVC, it was observed that the algorithm proved to be computationally expensive due to the fact that it encountered several crashes. Since there were not enough computational resources to execute the code, a sample was taken to serve as training for hyperparameter tuning.

Finally, according to the performance metrics and the confusion matrix, we suggest the use of Random Forest for predicting the types of cybersecurity attacks based on the predictors provided. This model returned an average recall of 99%, and performed well on predicting the values for minority classes.