Vernicia Gage, IO, EH

Project – Cyber Security Classification

Background

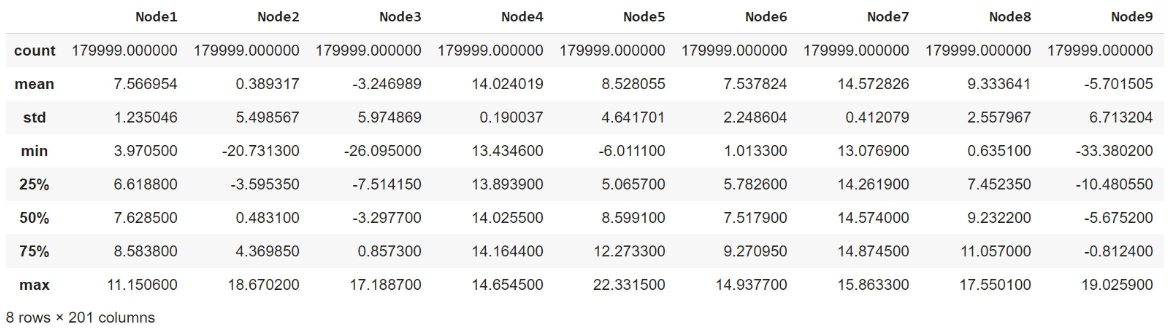
Use of data science and analytics to solve complicated problems has become one of the most pressing issues in recent times. All organizations generate data, and most organizations will store this data in some way and use it for simple analysis. As systems around the world become more interconnected, more data is being generated and information is being stored and this information needs to be protected.

This project analyzes a data set of user activity of different nodes on a computer network to predict if activity on the network is malicious or not. Analysis of the system and generation of an algorithm to predict whether this activity is malicious or not is done via four steps: using data visualization to analyze features, conduct data-preprocessing, generating models to predict the classification of the activity on the network, and deciding what method produces the best results and using said method to predict the classification of network activity as malicious or not.

Data Analysis

Descriptive Statistics

The provided training data set contained 180,000 observations and 200 predictors. In addition to these is a binary response variable that states whether the activity is malicious (1) or not (0). When viewing the descriptive statistics of the model, such as the mean, standard deviation, maximum and minimum values, it is observed that the data needs to be standardized in order to bring all of the value's standard deviations to 1. As most of the maximums of the values are within the same order of magnitude, it was decided that normalization was not necessary. Part of the descriptive statistics can be viewed in Figure 1.



	Node1	Node2	Node3	Node4	Node5	Node6	Node7	Node8	Node9
count	179999.000000	179999.000000	179999.000000	179999.000000	179999.000000	179999.000000	179999.000000	179999.000000	179999.000000
mean	7.566954	0.389317	-3.246989	14.024019	8.528055	7.537824	14.572826	9.333641	-5.701505
std	1.235046	5.498567	5.974869	0.190037	4.641701	2.248604	0.412079	2.557967	6.713204
min	3.970500	-20.731300	-26.095000	13.434600	-6.011100	1.013300	13.076900	0.635100	-33.380200
25%	6.618800	-3.595350	-7.514150	13.893900	5.065700	5.782600	14.261900	7.452350	-10.480550
50%	7.628500	0.483100	-3.297700	14.025500	8.599100	7.517900	14.574000	9.232200	-5.675200
75%	8.583800	4.369850	0.857300	14.164400	12.273300	9.270950	14.874500	11.057000	-0.812400
max	11.150600	18.670200	17.188700	14.654500	22.331500	14.937700	15.863300	17.550100	19.025900

8 rows × 201 columns

Figure 1: Descriptive statistics for Nodes 1 – 9

Data Visualization

Most of the activity on the system is not malicious, as shown by Figure 2. This is to be expected, as most activity on a network should be done by the average user and thus, would be benign.

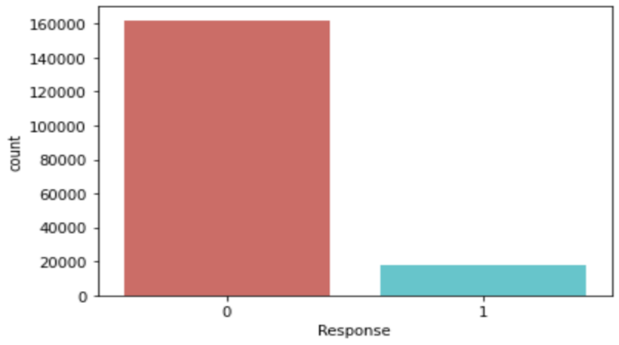


Figure 2: Bar graph showing Frequency of Response, with 0 being non-malicious and 1 being malicious.

Outliers for each of the nodes were also generated on a boxplot shown in Figure 3. For the sake of this first analysis of the data, it was decided to not remove the outliers and to see how the model and system responds and the level of accuracy that can be obtained.

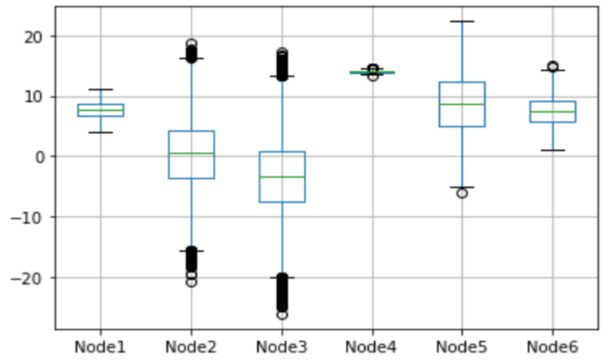
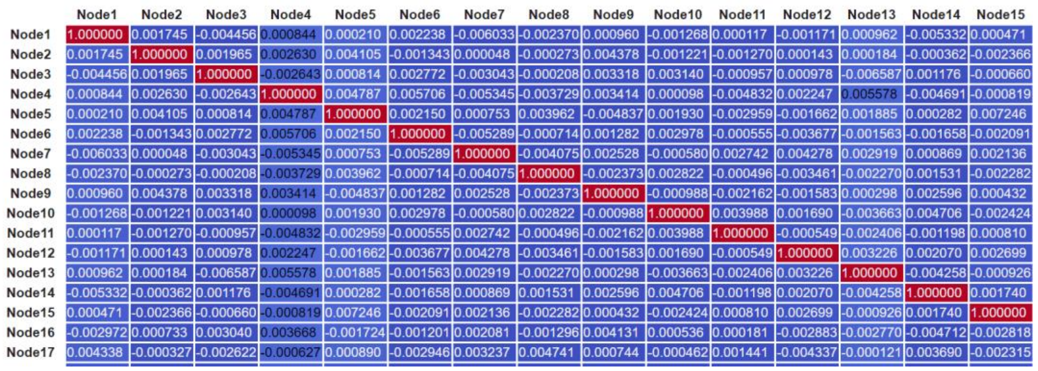


Figure 3: Boxplot of predictor outliers for Nodes 1 – 6

Finally, a correlation matrix was generated (Figure 4) to analyze whether the predictors were highly correlated or not. Due to the correlation values being low it was concluded that the predictors were not correlated and that we could proceed with the data pre-processing.



	Node1	Node2	Node3	Node4	Node5	Node6	Node7	Node8	Node9	Node10	Node11	Node12	Node13	Node14	Node15
Node1	1.000000	0.001745	-0.004456	0.000844	0.000210	0.002238	-0.006033	-0.002370	0.000960	-0.001268	0.000117	-0.001171	0.000962	-0.005332	0.000471
Node2	0.001745	1.000000	0.001965	0.002630	0.004105	-0.001343	0.000048	-0.000273	0.004378	-0.001221	-0.001270	0.000143	0.000184	-0.000362	-0.002366
Node3	-0.004456	0.001965	1.000000	-0.002643	0.000814	0.002772	-0.003043	-0.000208	0.003318	0.003140	-0.000957	0.000978	-0.006587	0.001176	-0.000660
Node4	0.000844	0.002630	-0.002643	1.000000	0.004787	0.005706	-0.005345	-0.003729	0.003414	0.000098	-0.004832	0.002247	0.005578	-0.004691	-0.000819
Node5	0.000210	0.004105	0.000814	0.004787	1.000000	0.002150	0.000753	0.003962	-0.004837	0.001930	-0.002959	-0.001662	0.001885	0.000282	0.007246
Node6	0.002238	-0.001343	0.002772	0.005706	0.002150	1.000000	-0.005289	-0.000714	0.001282	0.002978	-0.000555	-0.003677	-0.001563	-0.001658	-0.002091
Node7	-0.006033	0.000048	-0.003043	-0.005345	0.000753	-0.005289	1.000000	-0.004075	0.002528	-0.000580	0.002742	0.004278	0.002919	0.000869	0.002136
Node8	-0.002370	-0.000273	-0.000208	-0.003729	0.003962	-0.000714	-0.004075	1.000000	-0.002373	0.002822	-0.000496	-0.003461	-0.002270	0.001531	-0.002282
Node9	0.000960	0.004378	0.003318	0.003414	-0.004837	0.001282	0.002528	-0.002373	1.000000	-0.000988	-0.002162	-0.001583	0.000298	0.002596	0.000432
Node10	-0.001268	-0.001221	0.003140	0.000098	0.001930	0.002978	-0.000580	0.002822	-0.000988	1.000000	0.003988	0.001690	-0.003663	0.004706	-0.002424
Node11	0.000117	-0.001270	-0.000957	-0.004832	-0.002959	-0.000555	0.002742	-0.000496	-0.002162	0.003988	1.000000	-0.000549	-0.002406	-0.001198	0.000810
Node12	-0.001171	0.000143	0.000978	0.002247	-0.001662	-0.003677	0.004278	-0.003461	-0.001583	0.001690	-0.000549	1.000000	0.003226	0.002070	0.002699
Node13	0.000962	0.000184	-0.006587	0.005578	0.001885	-0.001563	0.002919	-0.002270	0.000298	-0.003663	-0.002406	0.003226	1.000000	-0.004258	-0.000926
Node14	-0.005332	-0.000362	0.001176	-0.004691	0.000282	-0.001658	0.000869	0.001531	0.002596	0.004706	-0.001198	0.002070	-0.004258	1.000000	0.001740
Node15	0.000471	-0.002366	-0.000660	-0.000819	0.007246	-0.002091	0.002136	-0.002282	0.000432	-0.002424	0.000810	0.002699	-0.000926	0.001740	1.000000
Node16	-0.002972	0.000733	0.003040	0.003668	-0.001724	-0.001201	0.002081	-0.001296	0.004131	0.000536	0.000181	-0.002883	-0.002770	-0.004712	-0.002818
Node17	0.004338	-0.000327	-0.002622	-0.000627	0.000890	-0.002946	0.003237	0.004741	0.000744	-0.000462	0.001441	-0.004337	-0.000121	0.003690	-0.002315

Figure 4: Correlation Matrix for Nodes 1 – 15

Data Preprocessing

The data was analyzed and checked for null values and all 27 rows with missing predictor values were removed. Then the data was split into X values and Y values and the X values of the data were standardized to bring all the predictor standard deviations to 1. The training data was then split via k-fold cross validation into a training data set and a testing data set, and models were generated. It was decided to use the same preprocessed data for the decision tree model as the sklearn library's decision tree models cannot process data with missing values. Also, standardization of data for a decision tree, while not helpful, will not impede the model and it would not affect the model to use the already preprocessed data.

Model Generation

Logistic Regression Model

Due to the nature of our data set and the response variables being binary, it was decided that the data would be fit to a logistic regression model. This was done by fitting the model with the partitioned training data set, generating predictions for the testing data set then those values were compared to the true values of the model testing data partitioned from the training data given. Then the k-folds resampling method with k = 5 was used to calculate the misclassification rate and the accuracy of the model and a Confusion Matrix (Figure 5) and Receiver Operating Characteristic (ROC) Curve (Figure 6) was generated.

Confusion Matrix for Logistic Regression Model		
	Predicted	
Actual	**0**	**1**
0	39912	564
1	3279	1230

Figure 5: Confusion Matrix for Logistic Regression Model

From the confusion matrix, the misclassification rate was calculated to be 0.08543 using the equation: $Misclassification\ Rate = \frac{FP+FN}{total}$. There is a sensitivity of 0.92 and a specificity of 0.68.

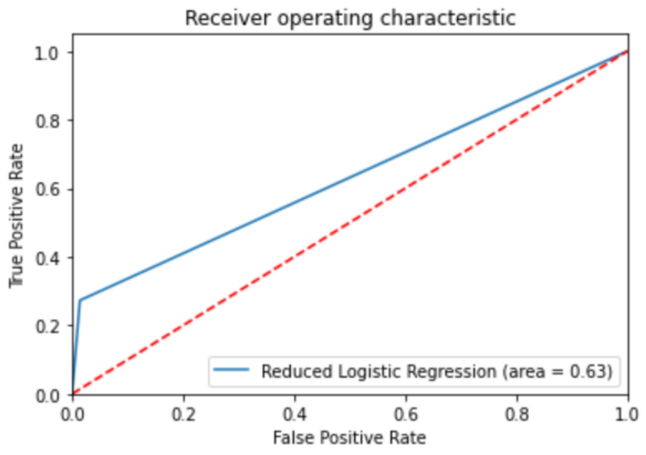


Figure 6: ROC (Receiver Operating Characteristic) Curve

The ROC curve is skewed to the top left corner and has a value of 0.63 which shows that it is more accurate than not, but there is room for model improvement.

Dimensionally Reduced Logistic Regression Model

It was then decided that the data may benefit from dimensional reduction of the model, as there are 200 predictors, and they may cause the model to overfit First an analysis was done to determine how much to dimensionally reduce the data this was done by using PCA analysis and testing dimensions from 0 to 200 and plotting the accuracy on a scatter plot. It was found that the accuracy for the training data did not decrease much, less than 0.01, with dimensional reduction.

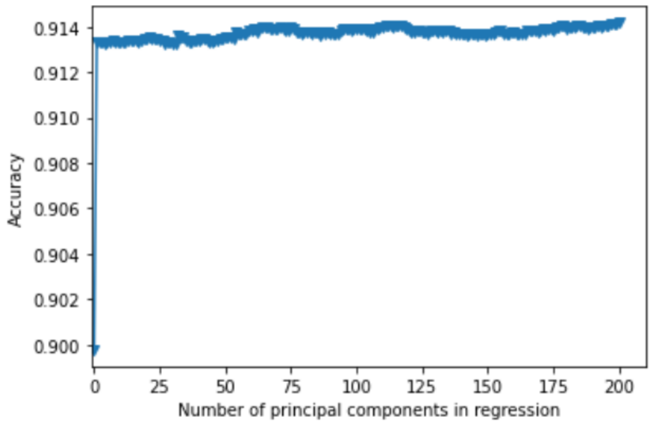


Figure 7: Scatter Plot of PCA Analysis

Then the model was fit with the partitioned training data set dimensionally reduced to 2 predictors, predictions for the testing data were generated then those values were compared to the true values of the model testing data partitioned from the training data given. Then the k-folds resampling method with k = 10 was used to calculate the misclassification rate and the accuracy of the model and a Confusion Matrix (Figure 8) and ROC Curve (Figure 9) was generated.

Confusion Matrix for Reduced Logistic Regression Model		
	Predicted	
Actual	**0**	**1**
0	39989	487
1	3353	1156

Figure 8: Confusion Matrix for Dimensionally Reduced Logistic Regression Model

From the confusion matrix, the misclassification rate was calculated to be 0.08536 using the equation: $Misclassification\ Rate = \frac{FP+FN}{total}$. There is a sensitivity of 0.92 and a specificity of 0.7.

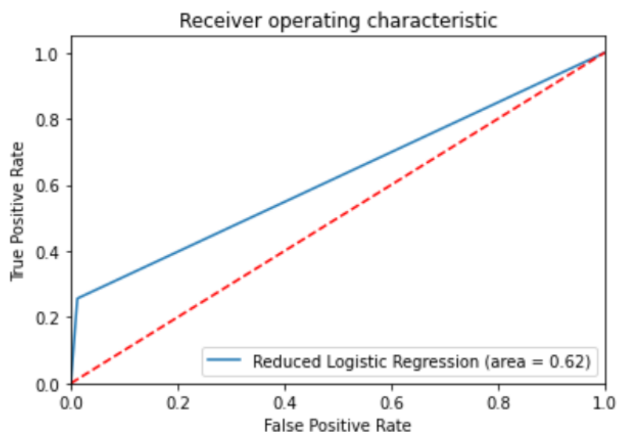


Figure 9: ROC Curve for Dimensionally Reduced Logistic Regression Model

Adaboost Classifier

Next, an Adaboost Classifier decision tree was generated. The standardized data was used as standardization does not help nor hinder the ability for decision trees to be accurate and the data had to be standardized for all other methods prior.

Then the model was fit with the partitioned training data set with 100 estimators then those values were compared to the true values of the model testing data partitioned from the training data given. Then the k-folds resampling method with k = 10 was used to calculate the misclassification rate and the accuracy of the model and a Confusion Matrix (Figure 10) and ROC Curve (Figure 11) was generated.

Confusion Matrix for Adaboost Classification Model		
	Predicted	
Actual	**0**	**1**
0	47706	807
1	4000	1472

Figure 10: Confusion Matrix for Adaboost Classification Model

From the confusion matrix, the misclassification rate, was calculated to be 0.08899 using the equation: $Misclassification\ Rate = \frac{FP+FN}{total}$. There is a sensitivity of 0.92 and specificity of 0.65.

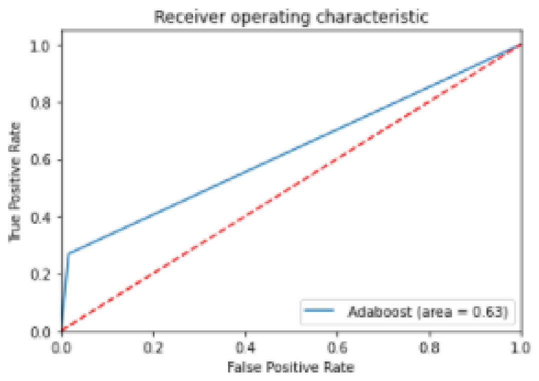


Figure 11: ROC Curve for Adaboost Classification Model

Linear Support Vector Classifier

It was decided that the data would be fit with a Support Vector Classifier (SVC). It was decided that a LinearSVC would be used in place of an SVC with a linear kernel as LinearSVC typically scales better with larger data sets. The model was fitted with the partitioned training data set, generating predictions for the testing data set then those values were compared to the true values of the model testing data partitioned from the training data given. Then the k-folds resampling method with k = 5 was used to calculate the misclassification rate and the accuracy of the model and a Confusion Matrix (Figure 12) and ROC Curve (Figure 13) was generated.

Confusion Matrix for LinearSVC Model		
	Predicted	
Actual	**0**	**1**
0	48303	268
1	4477	934

Figure 12: Confusion Matrix for LinearSVC Model

From the confusion matrix, the misclassification rate was calculated to be 0.08789 using the equation: $Misclassification\ Rate = \frac{FP+FN}{total}$. There is a sensitivity of 0.92 and specificity of 0.78.

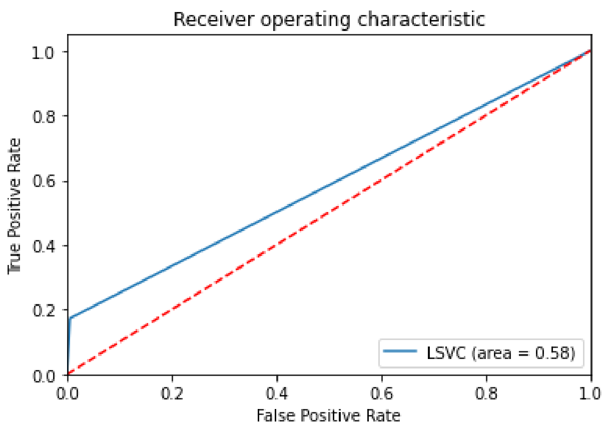


Figure 13: ROC Curve for LinearSVC

The ROC curve is skewed to the top left corner and has a value of 0.58.

Neural Network

Neural Network is a predictive analytic technique used for classification or prediction of a variable. This is a form of machine learning; it tries to predict values in a way a human brain would. We applied a neural network code for our project data, and we got the following results, shown in figure 14 and 15.

Confusion Matrix for Neural Network		
	Predicted	
Actual	**0**	**1**
0	44684	3826
1	3601	1871

Figure 14: Confusion Matrix for NN Model

From the confusion matrix, the misclassification rate was calculated to be 0.1375 using the equation: $Misclassification\ Rate = \frac{FP+FN}{total}$. There is a sensitivity of 0.92 and specificity of 0.32. The ROC curve has a value of 0.63.

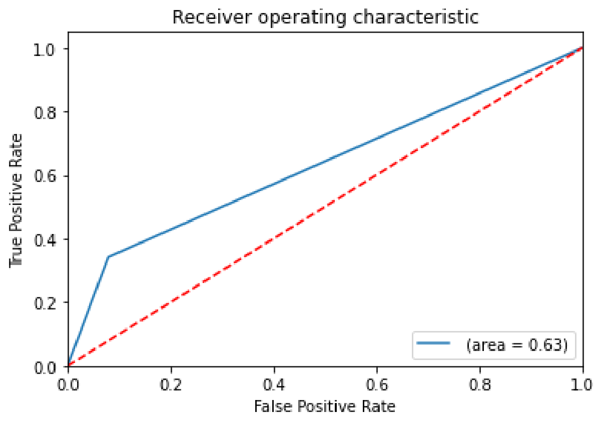


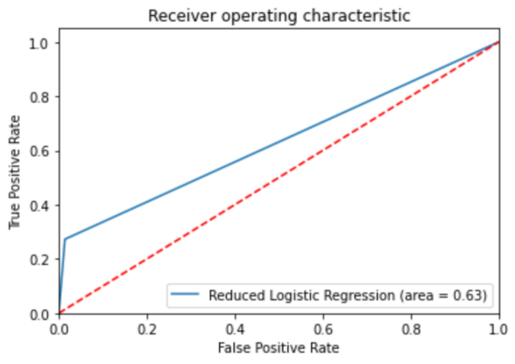
Figure 15: ROC Curve for NN

After running all the models with the unaltered training dataset, techniques were performed to try to increase the robustness of the model.

Model Robustness

Downsampling

Downsampling is the method of decreasing the majority class in the testing data set to have equal number of data points as the minority class in the testing set. There are many ways to calculate what points should be removed in downtown sampling, the approach chosen for this project was to do near miss downsampling. Near miss downsampling calculates the majority class points that are closest to the border between the majority class and the minority class and removes them. This helps the model performance by giving the model an equal number of data points for each class which in turn, removes some of the bias the model has towards classifying points as the majority class.



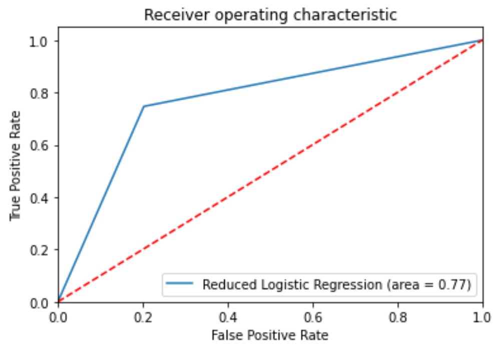


Figure 16: Side by side comparison of Logistic Regression ROC with normal(left) and downsampled(right) training data

When comparing model results of the original training set and the downsampled training set, the downsampled set performed much better. As seen in Figure 16, the logistic regression model ROC curve and for both the original training set and the downsampled training set are placed side by side, with the downsampled set having a better classification rate of the minority class. There's comparable model performance when comparing each model with the original trading set and with the downsampled training set, this was concluded that downsampling should be performed prior to fitting models. See appendix A for downsampled ROC curves.

Upsampling

Upsampling is the method that simulates or imputes additional data points to improve balance across classes. One way is to keep sampling cases from the minority class with replacement until the classes are balanced. Like downsampling, upsampling helps the model performance by giving an equal model of data. By giving the model an equal number of data points for each class, it improves model output by removing some of the model's bias toward classifying points as the majority class.

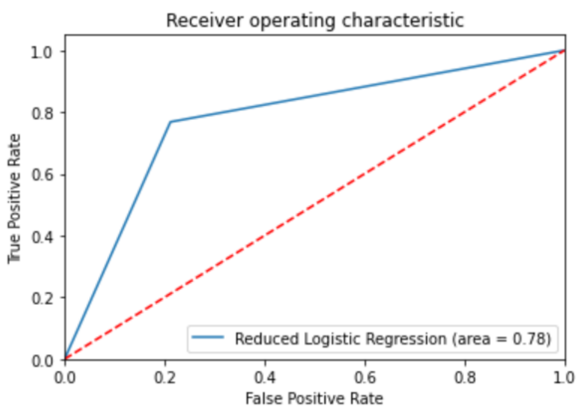


Figure 17: Logistic Regression ROC with upsampled training data

By the used of the ROC curve, we can observe that there was a better classification. See appendix B for upsampled ROC curves.

Conclusion

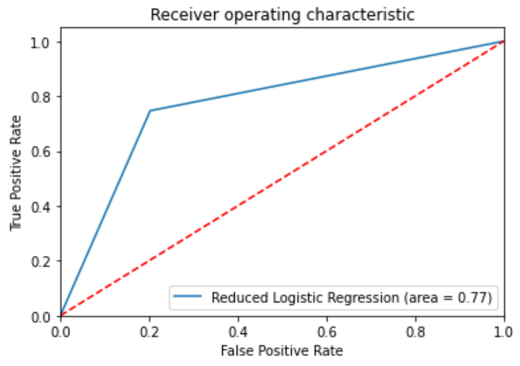
After analyzing the ROC curve and the misclassification rates of all the models tested in our data. We noticed that all models perform better after applying the downsampling method to it. This was concluded since in all models the number of true negatives increased, achieving our main focus of correctly classifying our minority class. It was decided to use the downsampling method with LinearSVC model to predict the Y values of the true testing data set. This model was chosen because it increases the true negatives correctly classified having a recall of 0.75 and a miss classification error rate of 0.20. As the misclassification error did not increase to a strong significant value, this method was deemed appropriate because the correct classification of the minority class did increase. Other than this, the down sampling LinearSVC gave a 77% true positive rate and 18% false positive rate for the minority class. Our model shows to have a linear tendency as overall, the linear models perform better than nonlinear models. This is another reason why we decided to predict the Y values with LinearSVC.

In the future the model can possibly be improved by using different classification models to fit and test the data, using subset selection to remove variables that have no effect on the system, or possibly replacing null values in the dataset with averaged values from other observations and removing outliers. Other possible techniques to improve current models are to try a combination of upsampling and downsampling such as SMOTETomek, dimensional reduction techniques shallow RF models and different nonlinear models such as SVM or KNN, to see how our model performs with these methods.

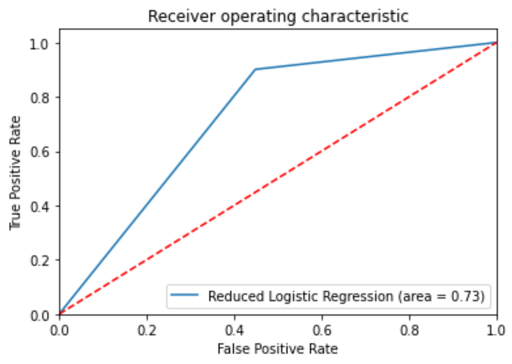
Appendices

Appendix A: Downsampled ROC Curves

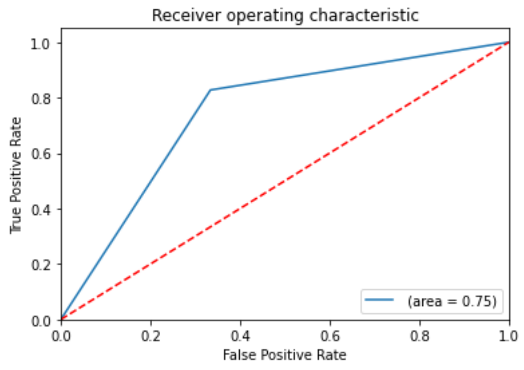
Logistic Regression



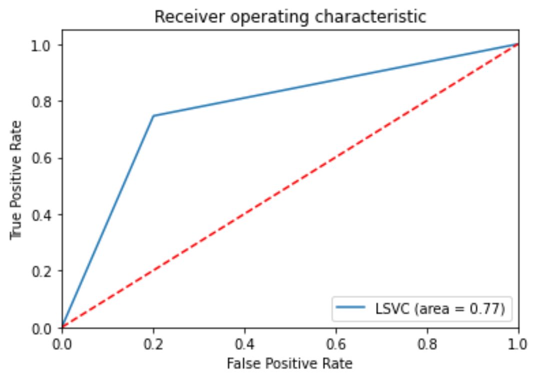
Reduced Logistic Regression



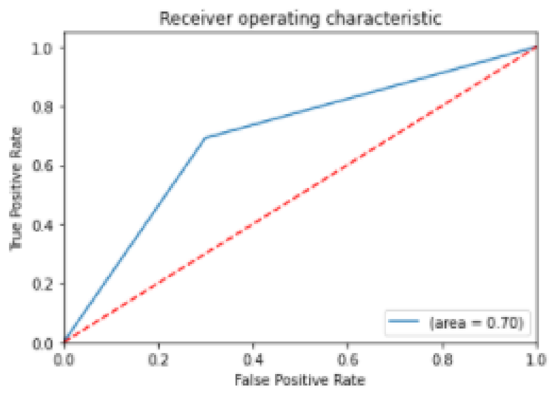
Adaboost



LinearSVC

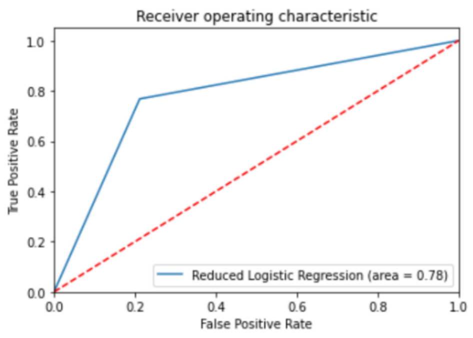


Neural Networks

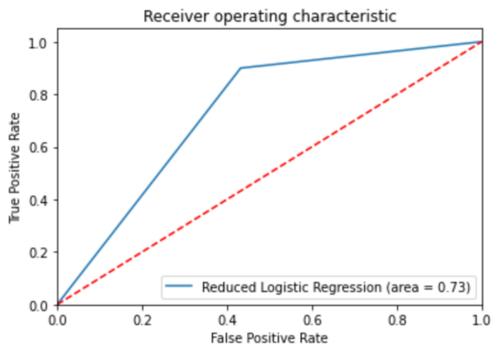


Appendix B: Upsampled ROC Curves

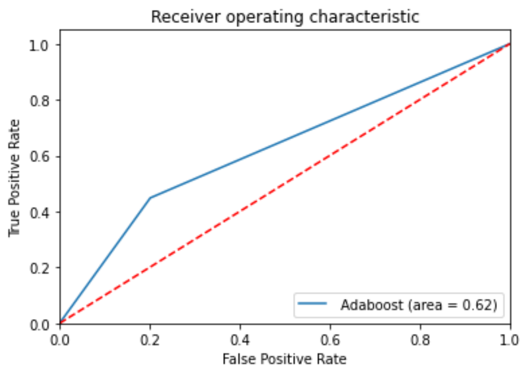
Logistic Regression



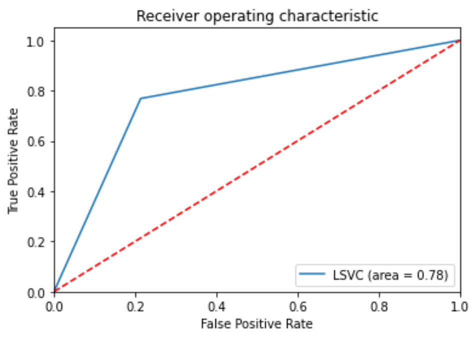
Reduced Logistic Regression



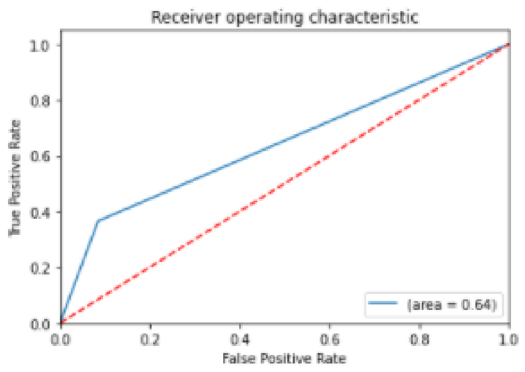
Adaboost



LinearSVC



Neural Networks


Receiver operating characteristic
True Positive Rate
False Positive Rate
(area = 0.64)