DePaul University College of Computing and Digital Media

Detecting Anomalous Network Activity in the KDD 1999 Data Set

CSC 424 – Advanced Data Analysis

Yanhong Simokat
7-18-2018

Executive Summary

"Many of the nation's essential and emergency services, as well as our critical infrastructure, rely on the uninterrupted use of the Internet and the communications systems, data, monitoring, and control systems that comprise our cyber infrastructure. A cyber-attack could be debilitating to our highly independent Critical Infrastructure and Key Resources (CIKR) and ultimately to our economy and national security"

Homeland Security Council, *National Strategy for Homeland Security*, 2007

Cyber threats are an ever-present problem in our modern technology driven lives. From the inconvenience of Facebook going offline to interfering with elections by malicious state actors and disruptions to core infrastructure the threat ranges from the mildly inconvenient to life and societal threatening. This means new and better means of detecting threats are critical to maintaining and securing our standards for our way of life.

The objective of this project is to develop a classifier that can accurately detect "good" and "bad" connections on a network. A successful detection methodology could be used to protect a network from hackers and malware. A well-studied open data set from the 1999 KDD Cup challenge is used to train a variety of models to determine which is best suited for the problem. The data set was generated by MIT's Lincoln Lab in 1998 for an Intrusion Detection Systems (IDS) challenge created by the Defense Advanced Research Projects Agency (DARPA).

An exploratory analysis of the data is undertaken to understand basic distributional and numerical qualities of the data. Several supervised and unsupervised learning techniques are applied to the data to determine the most accurate and most robust classifier. Five different logistic regression models are constructed. One containing the full set of features which is used primarily to set a baseline for comparison against all others. The second is a subset selected logistic regression using stepwise selection. Principal Component Analysis is used to explore the relationships in the features and for feature selection to compare the differences in selected features in another logistic regression. A subset of the Principal Component Analysis selected features is also used to create another logistic regression. Finally, a penalized logistic regression model is created to check the effect of shrinkage on the model. For additional comparison a linear discriminant analysis is also performed on the data. Finally, a Radom Forest model is also constructed as an alternative classifier to provide comparison against an ensemble method. A basic summary of the findings is summarized in the following table.

Figure 1. Summary results of all models (larger version in analysis)

Model	Accuracy	Accuracy 95% CI	Kappa	Mcnemar's Test p-value	Sensitivity	Specificity	Balanced Accuracy	# of Features	# High VIF Features	AIC
LDA	0.9858	(0.9853, 0.9862)	0.961	< 2.2e-16	0.9971	0.9508	0.974	37	N/A	N/A
Full Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9706	0.9837	37	10	48139
Stepwise Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9706	0.9837	34	9	48133
PCA Selected Features Logistic	0.9875	(0.9871, 0.988)	0.9659	< 2.2e-16	0.9969	0.9586	0.9777	16	3	64855
Modified PCA Selected Logistic	0.9876	(0.9871, 0.988)	0.966	< 2.2e-16	0.997	0.9584	0.9777	12	0	66262
GLMnet LASSO Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9704	0.9836	36	N/A	N/A
RandomForest	0.9995	(0.9994, 0.9996)	0.9987	< 2.2e-16	0.9999	0.9981	0.999	N/A	N/A	N/A

Depending on selection criteria there are multiple possibilities for the best model from the table. If selecting based on the best overall accuracy though comes from the Random Forest ensemble. Discussion of the analysis results is provided in the technical section on results.

While the modeling in this analysis has produced some interesting findings, it is important to denote that the problem in this analysis is a simplified version of the original data challenge. While this simplification does not detract from the results themselves it means different operating assumptions compared to other analyses on the same data. This means other results are not necessarily directly comparable. Also, since the originating data is the byproduct of a simulation it may not be as informative or applicable to the real world as one might hope. Reapplying the same techniques on a labeled data set from real networks might partially remediate this limitation but would need to be balanced against other limitations such as another network may not see certain attack types in the real world making the detector less sensitive to anomalies caused by them. The literature suggests this may not be a problem for certain methods like Random Forest but should still be validated in practice and indicates that machine learning based methods cannot fully replace other detection methods. They may be best suited in simply providing an additional layer of information and help sort prioritization. Finally, in our modeling problem the effect of the time series created by design in some of these features is not considered, this may not strictly be necessary since the features that come from temporally dependent data are from a fixed and predefined window, but nonetheless any temporal dependency that is not readily observable due to the features being calculated and not from raw data will likely affect the accuracy of a model that does not account for it.

Additional research and combining additional data sources and detection techniques is a promising way to provide substantive improvement over any individual method on its own. Thus, for security professionals the need to work holistically is key and why many businesses and governments are relying on the Cybersecurity Fusion Center model to provide an integrated approach to all aspects of cybersecurity.

ABSTRACT

Anomaly detection has numerous real-world applications. Perhaps one of the most significant, given the cornerstone technology plays in our society, is in the application of anomaly detection in intrusion detection on computer networks. The analysis reviews the literature for foundation and guiding research approach. An exploratory analysis on data is performed on the KDD Cup 1999 data set to provide a sense of the nature of the data. The analysis also reviews several classification algorithms and compares their accuracy with each other using a confusion matrix and discusses their relative complexities. Considerations of how this could be deployed in production and future research directions are also discussed.

INTRODUCTION

The objective of this project is to develop a classifier that can accurately detect "good" and "bad" connections on a network. A successful detection methodology could be used to protect a network from hackers and malware. A well-studied open data set from the 1999 KDD Cup challenge is used to train a variety of models to determine which is best suited for the problem. While the original challenge was to classify 22 different attack types and two types of normal connections. Attacks can be grouped into four categories which are denial of service, probing, unauthorized access to local super user, and unauthorized remote access. (Stolfo et al, 2000)

The data set was generated by MIT's Lincoln Lab in 1998 for an Intrusion Detection Systems (IDS) challenge created by the Defense Advanced Research Projects Agency (DARPA). The data set is a simulation of a typical United States Airforce (USAF) Local Area Network (LAN) configuration receiving a mixture of normal and malicious connection types. (Stolfo et al, 2000) The data is part of the University of California Irvine's Machine Learning Repository which is an online portal for free data sets available for researchers to learn applying techniques on well-studied data as well as for testing new techniques to compare against known ones. The data contains over 4.8 million observations and 42 features. Some features are basic TCP connection information, others are features computed by domain knowledge from subject matter experts in the security space, and others still are computed using a two-second time window. The data set is known to contain many duplicates which can inflate the accuracy of the models so the data in the analysis has been de-duplicated. The test data set is known to have very different distributional properties from the training data set and contains additional attack types not present in the training data set. It is also not labeled. Due to these constraints in this more simplified problem the test data set is split 75-25 for testing and training respectively.

The original simulated data comes pre-split into training and testing groups, but in the original KDD challenge the testing set has different distributional properties than the training data set. (Tavallaee, 2009) The testing data set also contains 14 additional attack types not in the training set. The testing set is also unlabeled. In order to address these limitations, the data is de-duplicated to prevent overestimating the model fits and also to make sure that there are no duplicates between a new training and testing split from the original training data. This data is split using 75%-25% for training and testing respectively. Since the data set even after de-duplication still contains over a million observations this is reasonable.

Other changes from the data as provided from the challenge to those used in the analysis include re-encoding the response variable into a binary variable where 0 is normal and 1 is abnormal. The data's original 42 features are reduced to 37 by eliminating non-numeric features and features that had no variance.

Tables defining the features in the data set are recreated from source. (Stolfo et al, 2000)

feature name	*description*	*type*
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Table 1: Basic features of individual TCP connections.

feature name	*description*	*type*
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

Table 2: Content features within a connection suggested by domain knowledge.

feature name	*description*	*type*
count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Traffic features computed using a two-second time window.

LITERATURE REVIEW

Since this is a well-studied data set there were numerous resources to choose from for inspiration on how to approach the data and problem in the challenge. The general suitability of machine learning and data mining to security problems is described in both Maloof's collection of articles (2006) as well as in Dua and Du (2011). Specifically, Maloof states that "For some detection problems in computer security, existing data mining and machine learning methods will suffice." (Maloof, 2006)

Specifically, to the suitability of detecting anomalous network activity various techniques have been applied in the literature. The appropriateness of data driven approaches to this problem is summarized as "Data mining can play a useful role in intrusion detection. In our research, we have pursued both classification and clustering approaches to see how they can be used to improve MITRE's network defense." (Bloedorn, 2006) A nice summary of numerous examples of data mining and machine learning for misuse/signature/anomaly detection is presented in Dua and Du (2011) spanning numerous cited analyses, the type of data used on them, and host versus network level scope. Other noteworthy papers on network anomaly detection include Lakhina et al's (2004) approach using PCA as well as Zhang and Zulkernine's (2006) using k-nearest neighbor, support vector machines, and random forest. Dua and Du specifically denote the suitability of random forest in that "The random forests algorithm has better predictive accuracy and efficiency on large data sets in high dimensional feature space. Network traffic flow has such data characteristics…" (2011) which justifies the decision to include it with the traditional statistical classifiers

Limitations due to temporal aspects are also known in the literature and for anomaly detection is referred to concept drift. "An important phenomenon for applications for computer security is concept drift… when examples have certain labels for periods of time and then have different labels for other periods of time." (Maloof, 2006) This makes an already difficult detection problem much harder. It also is difficult to account for in modeling since "Concept drift can occur quickly or gradually and on different time scales." (Maloof, 2006) Thus, for real or advanced applications factoring the effects of time and concept drift are important to ensuring a detector that is resilient to such pitfalls.

METHODS

Since the primary objective is developing an accurate classifier numerous technique are used to provide a detailed comparison against each other in order to select which model may be best fit from those created. The methods used are linear discriminant analysis, multiple variations of logistic regression, principal component analysis, and random forest.

A full logistic regression model was created primarily as a baseline for comparison against all other methods. Since the dependent variable was simplified to a binary categorical variable logistic regression is appropriate. The variations of logistic regression include stepwise selected logistic regression, LASSO logistic regression, and two logistic regression influenced by the results of a principal component analysis. These other approaches were attempted to try to get greater accuracy as well as to remediate issues with the models that are discussed later. A principal component analysis was also conducted to perform dimensionality reduction. Finally, since the literature suggested that random forest was an appropriate technique and that ensembles could outperform traditional models in this problem setting a random forest model was created and compared.

This analysis was performed using R version 3.5.0 released in April 2018. A variety of packages were used including corrplot, ggplot2, car, ggfortify, glmnet, caret, MASS, and H2O. H2O version is 3.20.0.2. The system the analysis was ran on was a Windows 10 desktop with a 6 core Intel i7 processor at 3.3Mhz with 128 gigabytes of physical memory.

Model	Accuracy	Accuracy 95% CI	Kappa	Mcnemar's Test p-value	Sensitivity	Specificity	Balanced Accuracy	# of Features	# High VIF Features	AIC
LDA	0.9858	(0.9853, 0.9862)	0.961	< 2.2e-16	0.9971	0.9508	0.974	37	N/A	N/A
Full Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9706	0.9837	37	10	48139
Stepwise Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9706	0.9837	34	9	48133
PCA Selected Features Logistic	0.9875	(0.9871, 0.988)	0.9659	< 2.2e-16	0.9969	0.9586	0.9777	16	3	64855
Modified PCA Selected Logistic	0.9876	(0.9871, 0.988)	0.966	< 2.2e-16	0.997	0.9584	0.9777	12	0	66262
GLMnet LASSO Logistic	0.9904	(0.99, 0.9907)	0.9738	< 2.2e-16	0.9968	0.9704	0.9836	36	N/A	N/A
RandomForest	0.9995	(0.9994, 0.9996)	0.9987	< 2.2e-16	0.9999	0.9981	0.999	N/A	N/A	N/A

Table 4 Summary of all model results using confusion matrix

DISCUSSION AND RESULTS

The following table is a quantile plot of the variables to provide a sense of their distribution.

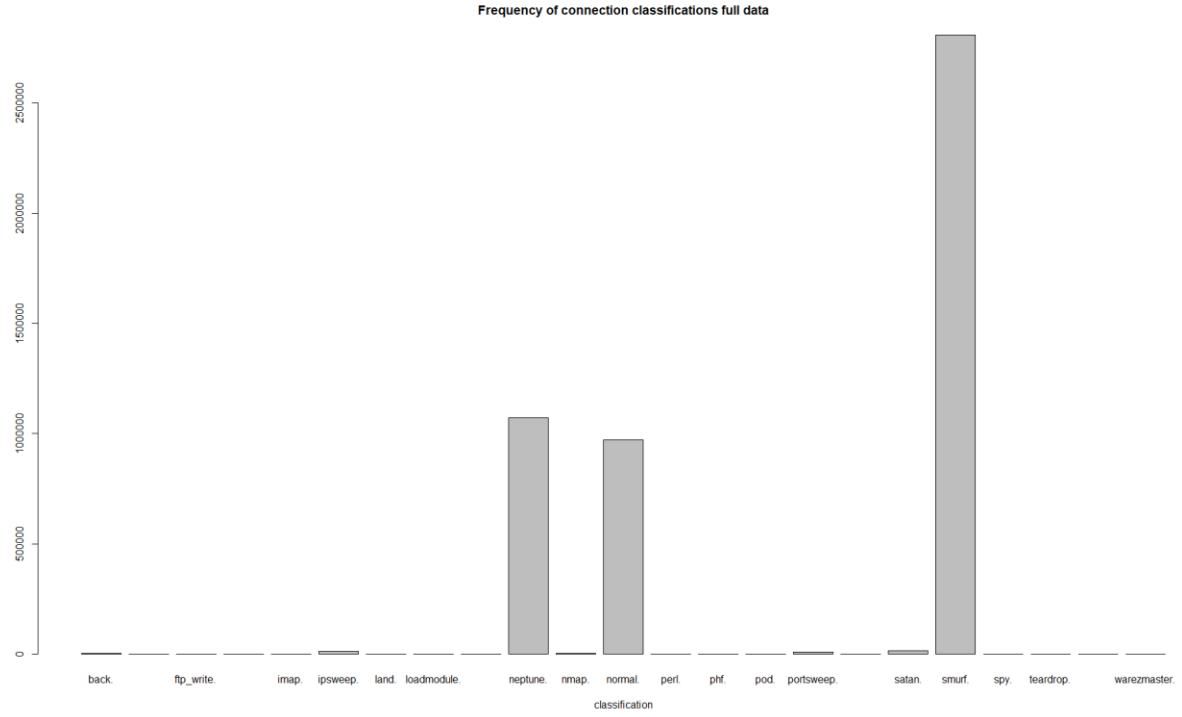
Table 5 Quantiles

	0.1%	1%	10%	25%	33.3%	50%	66.7%	75%	90%	99%	99.9%
durationQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4289.00	19085.00
src_bytesQuantiles	0	0.00	0.00	0.00	105.00	219.00	277.00	306.00	795.00	9178.00	61298.00
dst_bytesQuantiles	0	0.00	0.00	0.00	37.00	332.00	1035.00	1721.00	6065.00	30262.00	125015.00
landQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wrong_fragmentQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
urgentQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
hotQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	20.00
num_failed_loginsQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
logged_inQuantiles	0	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00
num_compromisedQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
root_shellQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
su_attemptedQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
num_rootQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	9.00
num_file_creationsQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
num_shellsQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
num_access_filesQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
is_host_loginQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
is_guest_loginQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
countQuantiles	1	1.00	1.00	2.00	3.00	8.00	17.00	34.00	209.00	294.00	511.00
srv_countQuantiles	1	1.00	1.00	2.00	3.00	7.00	12.00	15.00	24.00	125.00	511.00
serror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
srv_serror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
rerror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
srv_rerror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
same_srv_rateQuantiles	0	0.01	0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
diff_srv_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.07	0.75	1.00
srv_diff_host_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.40	1.00	1.00
dst_host_countQuantiles	1	1.00	12.00	50.00	87.00	224.00	255.00	255.00	255.00	255.00	255.00
dst_host_srv_countQuantiles	1	1.00	6.00	18.00	70.00	248.00	255.00	255.00	255.00	255.00	255.00
dst_host_same_srv_rateQuantiles	0	0.00	0.02	0.07	0.51	1.00	1.00	1.00	1.00	1.00	1.00
dst_host_diff_srv_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.04	0.06	0.08	0.78	1.00
dst_host_same_src_port_rateQuantiles	0	0.00	0.00	0.00	0.00	0.01	0.02	0.04	0.29	1.00	1.00
dst_host_srv_diff_host_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.05	0.25	0.67
dst_host_serror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00
dst_host_srv_serror_rateQuantiles	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00

The quantiles suggest there is skewness to the data which could influence the results since the logistic model has an underlying assumption of a binomial distribution. However, other methods like random forest do not make any specific distributional assumptions.

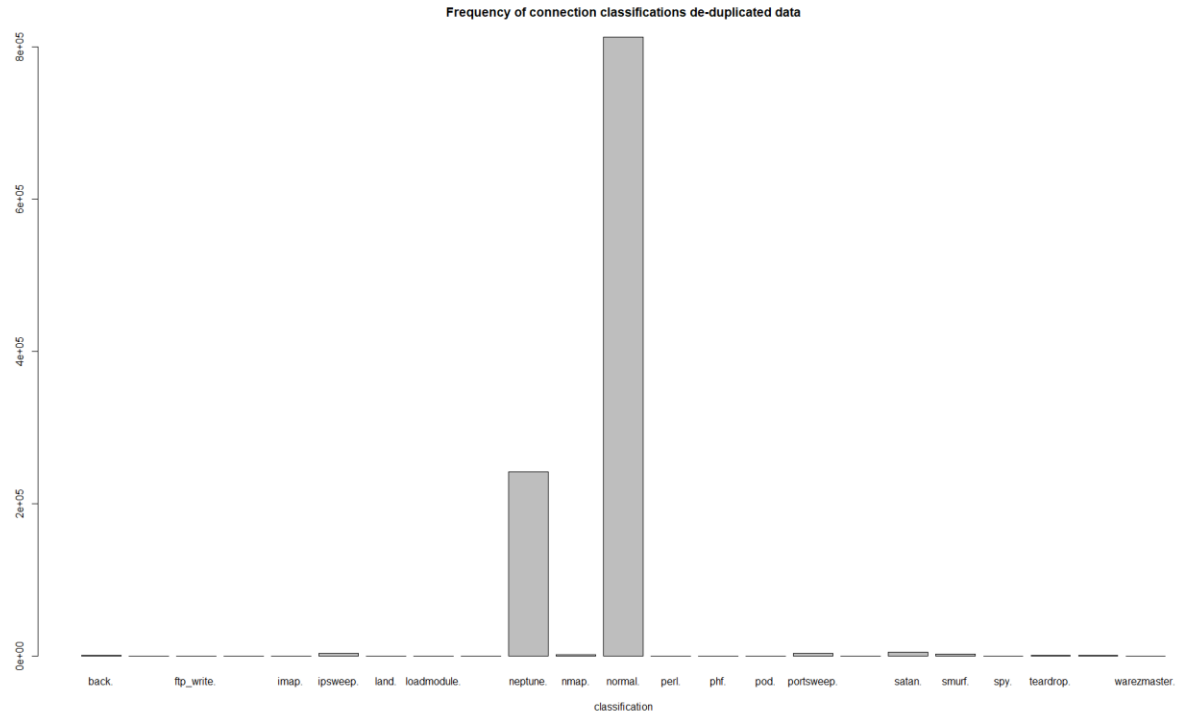
The following histogram shows the original distribution of the classification of all of the network traffic in the data set.

Figure 1 histogram of original classification distribution



The original distribution of the data shows that Neptune and smurf type attacks are very abundant and outnumber the normal cases in the data set. While this could be representative of a network under a heavy attack it becomes clear that the distribution of the classifications shifts dramatically once de-duplicated as shown in the next figure.

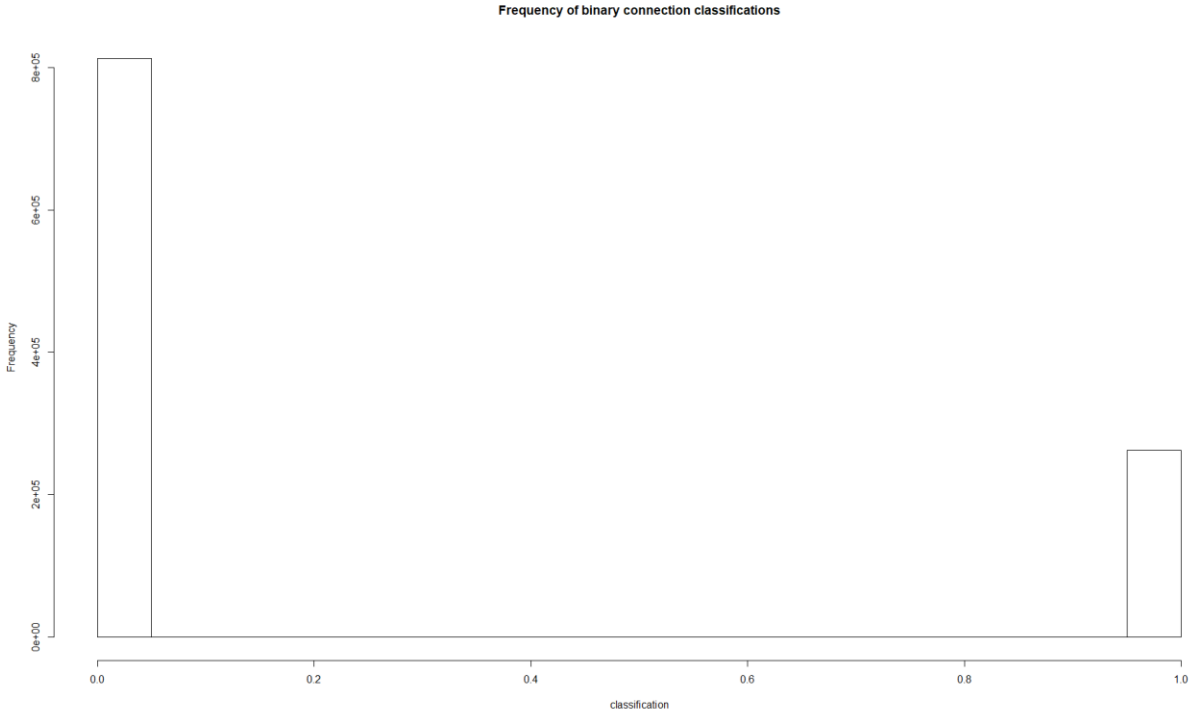
Figure 2 histogram of de-duplicated classification distribution



The de-duplicated histogram shows that normally classified traffic is the most abundant which seems more aligned with what one would expect in most cases. There are still an abundant concentration of Neptune attack examples in the distribution which may result in the model being more aware of how to detect Neptune type attacks over others.

Finally the distribution of the recoded classification is in figure 3.

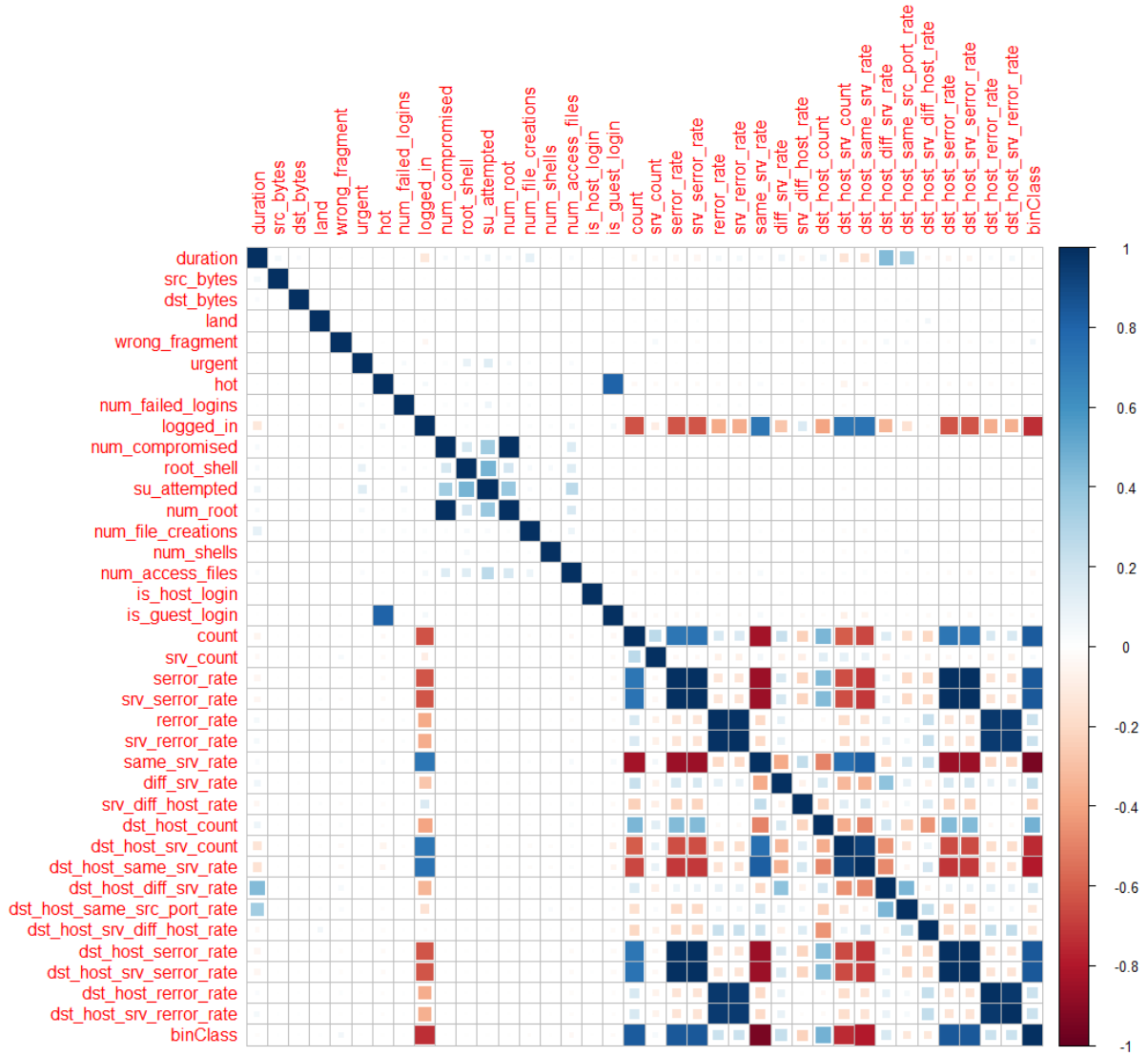
Figure 3 histogram of recoded classification



The histogram shows that the data is still mainly normal so there is some class imbalance between the cases. However there seem to be a fair number of abnormal observations in the data that it should be possible to proceed.

It is also important to take a look at the correlations between the features in the data set. This is depicted in the next figure.

Figure 4 correlation plot



The correlation plot shows there are several variables that are highly correlated with each other. This will be concerning when developing models since it puts a real risk for multicollinearity which is observed in the logistic regressions. Most of the other features have moderate or weak correlations so while variable selection and dimensionality reduction must be explored to remediate these strong relationships there are plenty of features still viable for models.

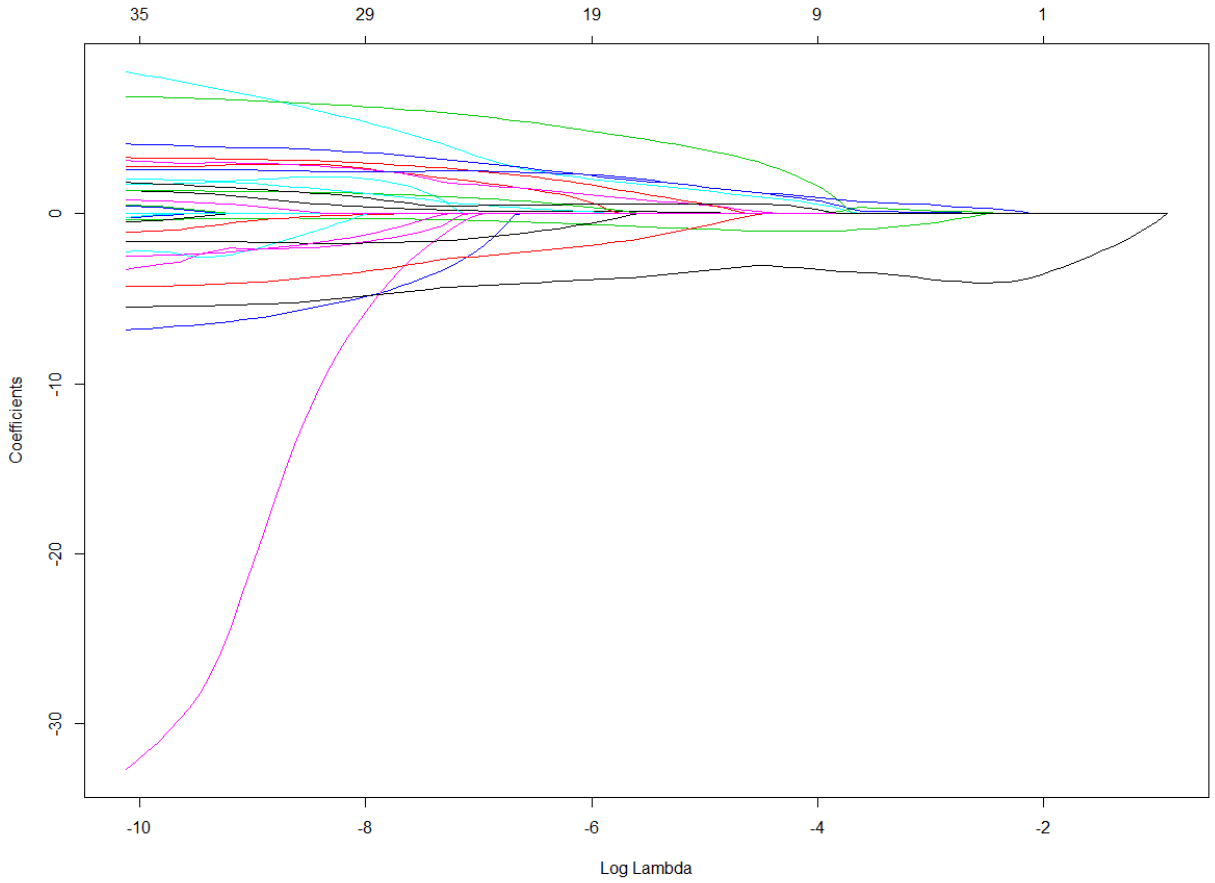
A logistic regression model was created using a full set of the features. Most coefficients in the model were significant but given the size of the data set this is mostly expected. The AIC and various attributes of the confusion matrix are summarized back in table 4, and while it is denoted that there are several features with VIF's greater than 10 they are not specified. The VIF's are summarized in the next table.

Table 6 VIF's for full logistic model

duration	src_bytes	dst_bytes	land
1.383100	1.023282	1.036020	1.038329
wrong_fragment	urgent	hot	num_failed_logins
1.000000	1.046503	142.711594	1.216693
logged_in	num_compromised	root_shell	su_attempted
1.973366	3.165485	1.108834	1.294152
num_root	num_file_creations	num_shells	num_access_files
2.854192	1.154165	1.040814	1.018553
is_host_login	is_guest_login	count	srv_count
1.000000	141.544655	10.296436	10.841204
serror_rate	srv_serror_rate	rerror_rate	srv_rerror_rate
10.816487	10.447474	87.788906	75.527481
same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count
5.078038	3.800994	1.548816	5.801921
dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate
3.364300	8.810185	4.584785	2.340172
dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate
1.955883	2.354068	2.795089	10.979128
dst_host_srv_rerror_rate			
31.044123			

These high VIF's indicate multicollinearity and require remediation. Multiple remediations were attempted such as stepwise selection procedure, PCA for dimensionality reduction (PCA discussion to come), and LASSO logistic regression. There were incremental improvements through the various techniques, but ultimately it took manually eliminating variables from the PCA reduced set to fully remediate the VIF in the features.

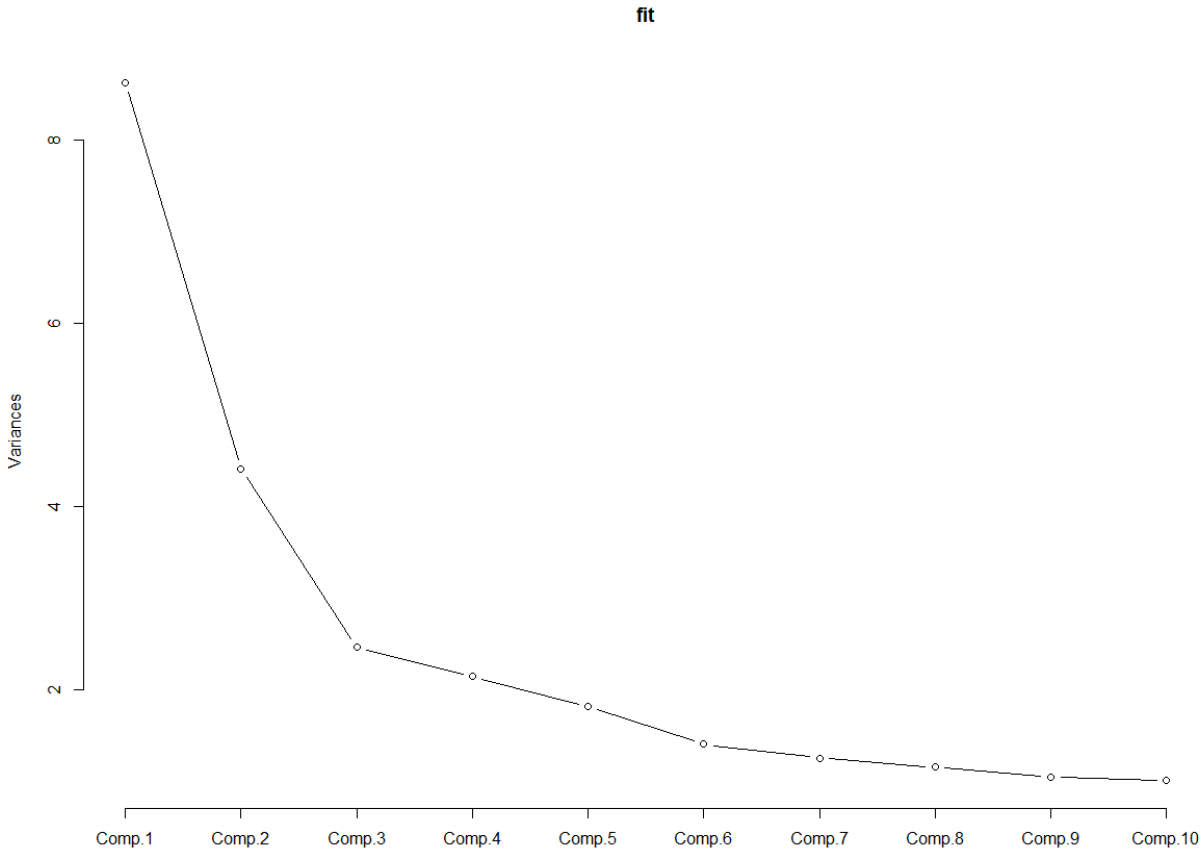
Figure 5 LASSO lambda plot



The GLMnet LASSO determined that optimal lambda was 4.036995e-05. While this model was more accurate than the other logistic regressions it retained almost all of the features and did not outperform the random forest model. So while an interesting result the other logistic regressions are reasonably close in their accuracy and easier to interpret their meaning.

A principal component analysis was conducted to perform variable selection. It took over 30 components to get to 100% of the variation. The scree plot in the next figure using the "knee bend" approach signals that the first two components are the most important.

Figure 6 PCA scree plot



The variables from the PCA and the loadings are in the next table.

Table 7 PCA selected variables

	Comp 1	Comp 2
duration	0	0
src_bytes	0	0
dst_bytes	0	0
land	0	0
wrong_fragment	0	0
urgent	0	0
hot	0	0
num_failed_logins	0	0
logged_in	-0.271	-0.154
num_compromised	0	0
root_shell	0	0
su_attempted	0	0
num_root	0	0
num_file_creations	0	0
num_shells	0	0
num_access_files	0	0
is_host_login	0	0
is_guest_login	0	0
count	0.286	0
srv_count	0	0
serror_rate	0.312	-0.137
srv_serror_rate	0.312	-0.137
rerror_rate	0	0.459
srv_rerror_rate	0	0.457
same_srv_rate	-0.327	0
diff_srv_rate	0.109	0
srv_diff_host_rate	0	0
dst_host_count	0.188	0
dst_host_srv_count	-0.28	0
dst_host_same_srv_rate	-0.299	0
dst_host_diff_srv_rate	0	0
dst_host_same_src_port_rate	0	0
dst_host_srv_diff_host_rate		0.157
dst_host_serror_rate	0.312	-0.137
dst_host_srv_serror_rate	0.313	-0.136
dst_host_rerror_rate	0	0.456
dst_host_srv_rerror_rate	0	0.455
binClass	0.321	0

The highlighted features are those whose values are different from zero. As previously mentioned these variables were put into a logistic regression model, but even this reduced model

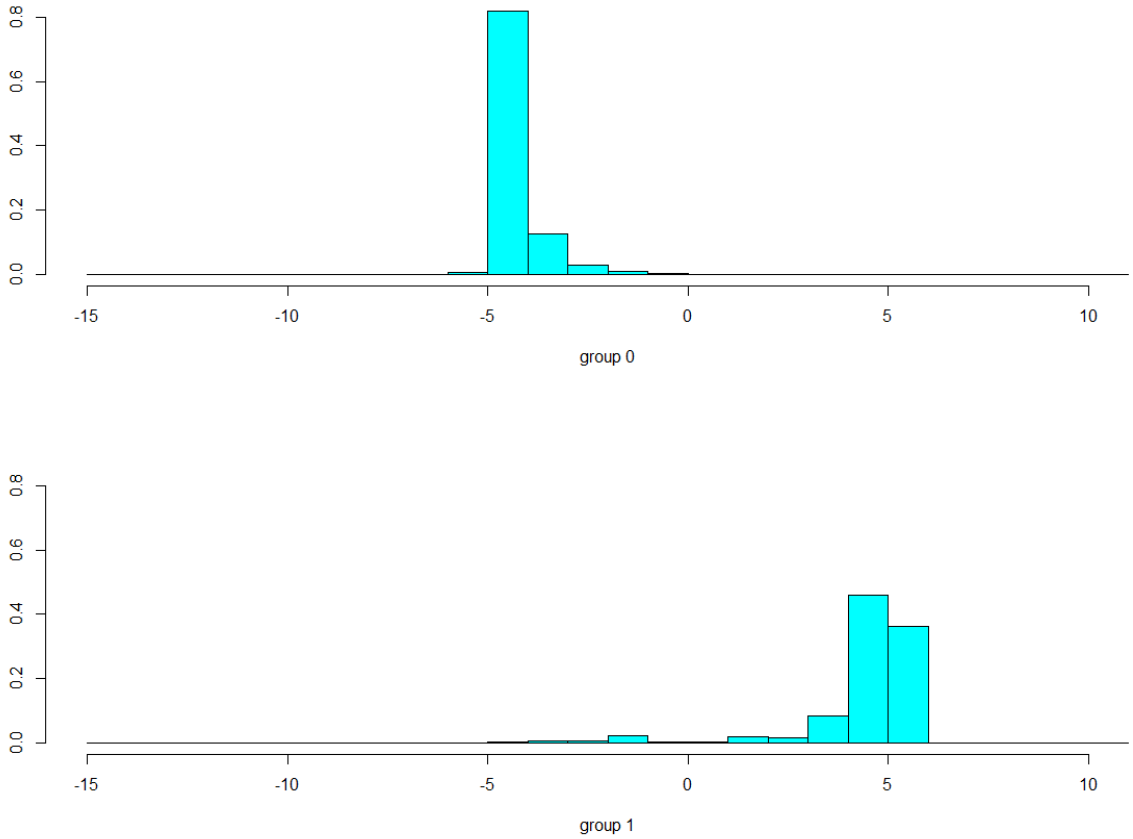
still had 3 features with VIF's greater than 10. After additional tuning 4 more features were removed resulting in a logistic regression that had no features with high VIF. The feature's and their VIF's in this last logistic regression were.

logged_in	count	serror_rate	same_srv_rate
1.253631	1.548420	2.343083	3.025882
diff_srv_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate
2.534945	4.476541	3.899968	5.886411
dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate
1.809184	2.050321	2.569664	1.483249

Given the simplicity of this model and its still high accuracy on the holdout set it may warrant further investigation as to its applicability in other analyses using similar data outside of anomaly detection. It may also be useful as a first layer of defense in a more thorough series of detectors since a logistic model can be scored in low latency using a variety of tools.

A linear discriminant analysis was also conducted. It performed the lowest overall in accuracy but is still highly accurate on the testing data set. This suggests that the data can be separated effectively which may be why in other literature (Dua and Du, 2011) kernel-based methods and neural network approaches are also effective.

Figure 7 LDA group distributions



The random forest model from H2O provided the highest classification accuracy out of all the models. In Zhang and Zulkernine (2006) their detection rate was 95% with a 1% false positive rate. The H2O random forest had a balanced accuracy of 99% and a false positive rate of 123/(11 + 65628) x 100% = .19% indicating that the model is more accurate and has a lower false positive rate.

CONCLUSION

The analysis shows that random forest provided the best predictive accuracy of the methods used while the PCA based logistic regression with the high VIF features dropped is the simplest to explain. For most applications in network intrusion detection a high degree of accuracy is preferable over ease of explanation, so the random forest method is preferred. Though in practice there may be considerations towards timing need to score where the simpler model may be easier to run in a real time streaming scoring engine for decisioning and alerting. The model developed in H2O appears to outperform the best-known model in the literature, but for this to be confirmed it would be important to rerun the analysis with the original classification labels to see if the accuracy remains so substantially improved compared to the published results. Additional methods for classification and anomaly detection should also be tried on the multi-label classification as well as on more realistic data would be directions for future analysis. While this model provides promise for applicability there are many hurdles to production deployment including mixing a version of the model trained on real network data not just a simulation. This in conjunction with signature-based detection methods can be effectively be used to mitigate threats more efficiently with existing security operations resources.

REFERENCES

Dua, S. and Du, X. (2011). *Data mining and machine learning in cybersecurity*. Boca Raton: CRC Press.

Maloof, M. A. (2006). Some Basic Concepts of Machine Learning and Data Mining (M. A. Maloof, Ed.). In *Machine learning and data mining for computer security* (pp. 23-43). London: Sprinnger.

Bloedorn, E. E., Talbot, L. M., & DeBarr, D. D. (2006). Data Mining Applied to Intrusion Detection: MITRE Experiences (M. A. Maloof, Ed.). In *Machine learning and data mining for computer security* (pp. 65-88). London: Springer.

Lakhina, A., Crovella, M., & Diot, C. (2004). Characterization of network-wide anomalies in traffic flows. *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement - IMC 04*. doi:10.1145/1028788.1028813

Zhang, J., & Zulkernine, M. (2006). Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. *2006 IEEE International Conference on Communications*. doi:10.1109/icc.2006.255127

Stolfo, S., Fan, W., Lee, W., Prodromidis, A., & Chan, P. (n.d.). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX00*. doi:10.1109/discex.2000.821515

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. doi:10.1109/cisda.2009.5356528

APPENDIX

```
# Project analysis
# Load the data
data <- read.csv("D:/Google
Drive/yanhong/DePaul/CSC424/Project/kddcup.data/kddcup.data.corrected")

labels <- c("duration",
"protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment",

"urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempte
d",

"num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_
host_login",

"is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_ra
te",

"same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
              "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",

"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror
_rate",
              "dst_host_srv_rerror_rate","classification")

names(data) <- labels

length(which(data$classification == "normal."))
# 972780
length(which(data$classification != "normal."))
# 3925650
# Check for missing data
sum(is.na(data))/prod(dim(data))
# 0 which indicates no missing data
colMeans(is.na(data))
# also 0 confirming it for all columns
# data set is known to have duplicates, let's remove them since they will effect the model
data.dedupe <- data[!duplicated(data),]
# Recheck normal/abnormal counts
length(which(data.dedupe$classification == "normal."))
# 812813
length(which(data.dedupe$classification != "normal."))
# 262178
# recode the response variable into a binary where 0 is normal and 1 is abnormal
data.dedupe$binClass <- ifelse(data.dedupe$classification == "normal.",0,1)
```

```
# Need numeric only for correlation plot
reddata <- data.dedupe[lapply(data.dedupe,class)!="factor"]
# Need numerics only with SD > 0 for correlation plot
reddata.delnovar <- reddata[lapply(reddata,sd)!= 0]

##################
# Feed into H2O
##################
library(h2o)
h2o.init()

# Import data into H2O
data.hex <- as.h2o(reddata.delnovar)

summary(data.hex)
quantile(x=data.hex)

data.split <- h2o.splitFrame(data=data.hex,ratios=.75)
data.train <- data.split[[1]]
data.test <- data.split[[2]]

## Convert H2OFrames into R data.frame
train.df <- as.data.frame(data.train)
test.df <- as.data.frame(data.test)

##################
# EDA
##################

# Distribution of the response variable before de-duplication
barplot(table(data$classification), main="Frequency of connection classifications full data",
      xlab="classification")
# Distribution of the response variable after de-duplication
barplot(table(data.dedupe$classification), main="Frequency of connection classifications de-
duplicated data",
      xlab="classification")
# Distribution of recoded response variable
hist(reddata.delnovar$binClass, main="Frequency of binary connection classifications",
    xlab="classification")

# Check the correlation plot
library(corrplot)
c = cor(reddata.delnovar)
corrplot(c, method="square")
# some signs of strong correlations off of the main diagonal so risk of multicolinearity
# also some factors are strongly correlated with the resposne variable
```

```
##############################################################################
# Try PCA
##############################################################################
# PCA_Plot functions
##############################################################################

PCA_Plot = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))
  p + geom_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names, colour = .names,
fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")
}

PCA_Plot_Secondary = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = data.frame(pcaData$rotation, .names = row.names(pcaData$rotation))
  p + geom_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names, colour = .names,
fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4")
}

PCA_Plot_Psyc = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = as.data.frame(unclass(pcaData$loadings))
  s = rep(0, ncol(loadings))
  for (i in 1:ncol(loadings))
```

```
  {
    s[i] = 0
    for (j in 1:nrow(loadings))
      s[i] = s[i] + loadings[j, i]^2
    s[i] = sqrt(s[i])
  }

  for (i in 1:ncol(loadings))
    loadings[, i] = loadings[, i] / s[i]

  loadings$.names = row.names(loadings)

  p + geom_text(data=loadings, mapping=aes(x = PC1, y = PC2, label = .names, colour = .names,
fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC1", y = "PC2")
}

PCA_Plot_Psyc_Secondary = function(pcaData)
{
  library(ggplot2)

  theta = seq(0,2*pi,length.out = 100)
  circle = data.frame(x = cos(theta), y = sin(theta))
  p = ggplot(circle,aes(x,y)) + geom_path()

  loadings = as.data.frame(unclass(pcaData$loadings))
  s = rep(0, ncol(loadings))
  for (i in 1:ncol(loadings))
  {
    s[i] = 0
    for (j in 1:nrow(loadings))
      s[i] = s[i] + loadings[j, i]^2
    s[i] = sqrt(s[i])
  }

  for (i in 1:ncol(loadings))
    loadings[, i] = loadings[, i] / s[i]

  loadings$.names = row.names(loadings)

  print(loadings)
  p + geom_text(data=loadings, mapping=aes(x = PC3, y = PC4, label = .names, colour = .names,
fontface="bold")) +
    coord_fixed(ratio=1) + labs(x = "PC3", y = "PC4")
}
```

```
#############################
# Modeling in R
#############################
library("car")

# try building a simple logistic regression model
logit <- glm(binClass ~ ., data=train.df, family="binomial")
summary(logit)
vif(logit)
# Try to predict the class on test data
p.logit = predict(logit, newdata=test.df[,1:37], type="response")
p.logit.round <- round(p.logit)

# Compare the results of the prediction
table(p.logit.round, test.df$binClass)
confusionMatrix(as.factor(p.logit.round),reference = as.factor(test.df$binClass))

# Subset selection using stepwise
logit.step <- step(logit)
steplogit.formula <- formula(logit.step)
logit.step.model <- glm(steplogit.formula, data=train.df, family="binomial")
summary(logit.step.model)
vif(logit.step.model)
# Try to predict the class on test data
p.step = predict(logit.step.model, newdata=test.df[,1:37], type="response")
p.step.round <- round(p.step)

# Compare the results of the prediction
table(p.step.round, test.df$binClass)
confusionMatrix(as.factor(p.step.round),reference = as.factor(test.df$binClass))


# how many PCA components
dataPCA = prcomp(reddata.delnovar)
summary(dataPCA)
round(dataPCA$rotation, 2)
plot(dataPCA)
PCA_Plot(dataPCA)
loadings(dataPCA)

dataPCA2 = prcomp(reddata.delnovar, scale=T)
summary(dataPCA2)
round(dataPCA2$rotation, 2)
plot(dataPCA2)
PCA_Plot(dataPCA2)
```

```
# PCA
scoresData <- dataPCA2$x
library("ggfortify")
autoplot(prcomp(reddata.delnovar), data = reddata.delnovar, label=F, label.size=3, loadings=T,
loadings.label=T,
        loadings.label.size=3)

# Pricipal Components Analysis
# entering raw data and extracting PCs
# from the correlation matrix
fit <- princomp(reddata.delnovar, cor=TRUE)
summary(fit) # print variance accounted for
loadings(fit) # pc loadings
plot(fit,type="lines") # scree plot
fit$scores # the principal components
#biplot(fit)

## Logistic regression based on PCA reduced variables from first two components
logit.reduced <- glm(binClass ~ logged_in + count + serror_rate + srv_serror_rate + rerror_rate +
srv_rerror_rate +
                same_srv_rate + diff_srv_rate + dst_host_count + dst_host_srv_count +
dst_host_same_srv_rate +
                dst_host_srv_diff_host_rate + dst_host_serror_rate + dst_host_srv_serror_rate +
                dst_host_rerror_rate + dst_host_srv_rerror_rate, data=train.df, family="binomial")
summary(logit.reduced)
vif(logit.reduced)
# Try to predict the class on test data
p.logit2 = predict(logit.reduced, newdata=test.df[,1:37], type="response")
p.logit.round2 <- round(p.logit2)

# Compare the results of the prediction
table(p.logit.round2, test.df$binClass)
confusionMatrix(as.factor(p.logit.round2),reference = as.factor(test.df$binClass))

## Logistic regression based on PCA reduced variables from first two components
## Three features still showed up with high VIF manually removing them and re-running the
model
## High VIF features are rerror_rate, srv_rerror_rate, and dst_host_srv_rerror_rate
logit.reduced2 <- glm(binClass ~ logged_in + count + serror_rate +
                same_srv_rate + diff_srv_rate + dst_host_count + dst_host_srv_count +
dst_host_same_srv_rate +
                dst_host_srv_diff_host_rate + dst_host_serror_rate + dst_host_srv_serror_rate +
                dst_host_rerror_rate, data=train.df, family="binomial")
summary(logit.reduced2)
vif(logit.reduced2)
# Try to predict the class on test data
```

```
p.logit3 = predict(logit.reduced2, newdata=test.df[,1:37], type="response")
p.logit.round3 <- round(p.logit3)

# Compare the results of the prediction
table(p.logit.round3, test.df$binClass)
confusionMatrix(as.factor(p.logit.round3),reference = as.factor(test.df$binClass))


## LASSO regression with glmnet
library(glmnet)
lassoglmmod <- glmnet(lasso.train <- sapply(train.df[,-38], as.numeric),
y=as.factor(train.df$binClass), alpha=1, family="binomial")
plot(lassoglmmod, xvar="lambda")
summary(lassoglmmod)
names(lassoglmmod)
print(lassoglmmod)
min(lassoglmmod$lambda)
which(lassoglmmod$lambda == min(lassoglmmod$lambda))
coef(lassoglmmod, s=4.036995e-05)
lasso.pred <- predict(lassoglmmod, lasso.test <- sapply(test.df[,-38], as.numeric), s=4.036995e-
05, type="class")
p.glmnet.round <- round(as.numeric(lasso.pred))

# Compare the results of the prediction
table(p.glmnet.round, test.df$binClass)
confusionMatrix(as.factor(p.glmnet.round),reference = as.factor(test.df$binClass))


## linear discriminant analysis
library(caret)
library(MASS)
trainLDA = lda(binClass ~ ., data=train.df)
trainLDA
plot(trainLDA)

# Try to predict the class on test data
p.test = predict(trainLDA, newdata=test.df[,1:37])$class
p.test

# Compare the results of the prediction
table(p.test, test.df$binClass)
confusionMatrix(p.test,reference = as.factor(test.df$binClass))
```

```
##########################
# H2O part of the analysis
##########################

# Set predictor and response variables
Y <- "binClass"
X <-
c("duration","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot","num_failed_login
s","logged_in",

"num_compromised","root_shell","su_attempted","num_root","num_file_creations","num_shells
","num_access_files",

"is_host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate
",

"srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_ho
st_srv_count",

"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv
_diff_host_rate",

"dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_ra
te")

# h2o RandomForest
rf.mod <- h2o.randomForest(x=X,y=Y,data.train, validation_frame =data.test)
summary(rf.mod)
rf.mod@model$validation_metrics
h2o.hit_ratio_table(rf.mod,valid = T)[1,2]
rf.pred <- predict(rf.mod, data.test)
#h2o.confusionMatrix(rf.pred, data.test)
# Workaround for H2O confusion matrix
tmp <- round(as.data.frame(rf.pred))
p.rf.round <- as.numeric(unlist(tmp))
# Compare the results of the prediction
table(p.rf.round, test.df$binClass)
confusionMatrix(as.factor(p.rf.round),reference = as.factor(test.df$binClass))
```