IMPROVEMENTS ON KDD'99 CUP RESULTS BY USING HEURISTIC RULES AND IMPROVED DATA SETS

by

Dillon Welch

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Science

HONORS PROGRAM
LOUISIANA TECH UNIVERSITY

May 2013

ABSTRACT

The KDD'99 Cup data set is widely used in signature detection, anomaly detection, and Intrusion Detection System (IDS) development. While it is a popular data set, it has several flaws. For results on this data set to be meaningful, steps to correct these flaws must be taken. One method is to use a modified data set, such as NSL-KDD which removes duplicates and contains records in proportion to their difficulty of classification. A common problem in machine learning is rare event detection, and one strategy for boosting results on rare events is to develop heuristic rules for them. Results, especially on rare events, are improved by using a combination of the NSL-KDD modified data set and heuristic rules for particular Remote to Local (R2L) attacks. In particular, when using the NSL-KDD training and testing sets the average cost decreased by 0.1201, the precision for R2L increased by 10.18%, and the recall for R2L increased by 28.22% when combining KNN with the heuristic rules as compared to just using KNN. This shows that results for rare event detection can be increased by taking into account the flaws of the data set and the patterns of the rare events when building a classification model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The KDD Cup '99 data set has been one of the most widely used data sets in signature detection, anomaly detection, and IDS development. It is based off of the 1998 DARPA Intrusion Detection Evaluation Program data set, which was developed as a simulation of a military network LAN peppered with attacks of four varieties: probing, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L). The types of attacks for each category can be seen in Table 1.

Table 1: Information on the Four Attack Categories

Category	**Description**	**Name**
Probing	The attacker scans network ports to find information and potential vulnerabilities	portsweep, ipsweep, nmap, satan, **saint**, **mscan**
Denial of Service (DoS)	The attacker overwhelms the network such that it is unable to handle legitimate traffic.	neptune, smurf, pod, teardrop, land, back, apache2, **udpstorm, processtable, mailbomb**
User to Root (U2R)	The attacker starts as a regular user and takes advantage of a vulnerability to gain root access on a system.	buffer_overflow, loadmodule, perl, rootkit, **xterm, ps, httptunnel, sqlattack, worm, snmpguess**
Remote to Local (R2L)	The attacker illegitimately gains local access to a machine on the network from a remote machine.	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, ware, spy, **snmpgetattack, named, xlock, xsnoop, sendmail**

The attack names that are in bold appear only in the testing set, the rest appear in both the training and testing sets. There are 39 total attack types, of which 16 are unique to the testing set. Data taken from [1].

The training data has 4,898,432 records, and the testing data has 311,027 records. The test data does not have the same probability distribution as the training data. There are three main feature groups in the KDD data. The first group is that of basic TCP features, detailed in Table 2. The second group contains content features, an example of which is the number of failed login attempts. This group is where the U2R and R2L attacks distinguish themselves. The third group has traffic features, which are calculated over a time interval of two seconds [1]. The features for the second and third groups are detailed on the next page in Tables 3 and 4.

Table 2: Basic TCP Feature Information.

Feature Name	**Description**	**Numeric Type**
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, for example TCP	discrete
service	network service on the destination, for example HTTP	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, and 0 otherwise	discrete
wrong_fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous

These features are attributes standard to all TCP/IP connections. Data taken from [1].

Table 3: Content Features Information.

Feature Name	Description	Numeric Type
hot	number of “hot” indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in, and 0 otherwise	discrete
num_compromised	number of “compromised” conditions	continuous
root_shell	1 if root shell is obtained, and 0 otherwise	discrete
su_attempted	1 if “su root” command is attempted, and 0 otherwise	discrete
num_root	number of “root” accesses	continuous
num_file_creations	the number of file creation operations	continuous
num_shells	the number of shell prompts	continuous
num_access_files	the number of operations on access control files	continuous
num_outbound_cmds	the number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the “hot” list, and 0 otherwise	discrete
is_guest_login	1 if the login is a “guest” login, and 0 otherwise	discrete

These features are values in a connection that are implied from domain knowledge. Data taken from [1].

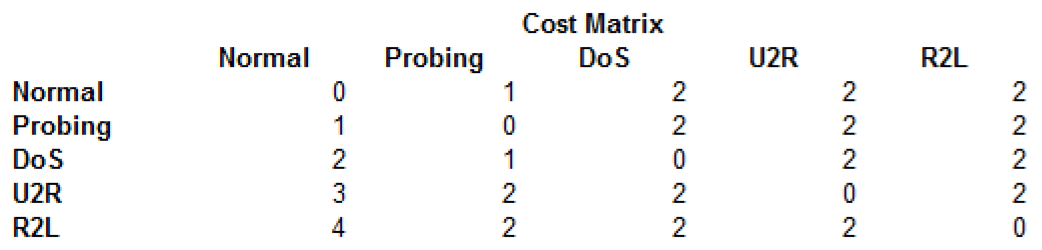
Table 4: Computed Traffic Features

Feature Name	Description	Numeric Type
count	the number of connections to the same host as the current connection in the past two seconds	continuous
serror_rate	% of connections that have same-host “SYN” errors	continuous
rerror_rate	% of connections that have same-host “REJ” errors	continuous
same_srv_rate	% of connections to the same service from the same host	continuous
diff_srv_rate	% of connections to different services from the same host	continuous
srv_count	the number of connections to the same service as the current connection in the past two seconds	continuous
srv_serror_rate	% of connections that have same-service “SYN” errors	continuous
srv_rerror_rate	% of connections that have same-service “REJ” errors	continuous
srv_diff_host_rate	% of connection to different hosts	continuous

Traffic features are calculated over an interval of two seconds and split into two groups. Same host features share a destination host with the current connection. Same service features share the service of the current connection. For attacks that operate over intervals larger than two seconds, these features are recalculated over a window of 100 connections and called connection-based features instead [1].

For the 1999 KDD Cup competition results, the web site detailing the methods of the winning entry was unavailable, but the second place entry used a tool called Kernel Miner and the third place entry used a mix of association rules and two sets of multiple decision trees [2]. Simple methods worked well though, and the 10th place participant scored fairly well using a 1-nearest neighbor approach.

The competition used a cost matrix to score the various entries, shown in Figure 1. To calculate the average classification cost, multiply each entry in the cost matrix with the entry in the same spot in the confusion matrix from the results. According to the results report, an even simpler method would have been to classify all test examples as a DoS[1] attack. This would have resulted in an average classification cost of 0.5220, which is better than the worst result submitted with an average cost of 0.9414 [3].



Cost Matrix

	Normal	Probing	DoS	U2R	R2L
Normal	0	1	2	2	2
Probing	1	0	2	2	2
DoS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

Figure 1: Cost matrix used to score the original results. Columns correspond to categories chosen by the classifier, and rows correspond to the correct classification. The lower total cost, the better [3].

Unfortunately, there are multiple flaws in the data set that skew any testing done on them. The first problem is that there is a large amount of duplicate records in both the training and testing sets, with the training set having 78% duplication and the testing set

1 In [3], it is erroneously stated to be probing instead of DoS. The error contradicts an earlier paragraph that states that “The cost matrix says that the cost incurred by classifying all examples as “probe” is not much over 1.0” and in fact the value is 1.039.

having 75% duplication. For more details, see Tables 6 and 7. This means that any learner will inevitably be skewed towards the duplicate records, which in this case are mainly probing and DoS attacks. DoS attacks alone are 71% of the testing set, see Table 5. It also prevents learners from effectively detecting rarer attacks such as U2R and R2L attacks which are much more harmful. The second problem is that the data set is very large. When dealing with large data sets, researchers may opt to use random sampling instead of using the entire data set. This is often driven from a resources standpoint; in many cases the data set may not fit in memory[2] or it may not be feasible time-wise to run algorithms on millions of records. Especially since this data is over 10 years old, sometimes it may have been as simple as available hard drive space. To make the problem worse, random sampling is usually justified by the random distribution of the data. In this data set though, the attack types are very unevenly distributed throughout the data set, for example the 4th, 5th, 6th, and 7th 10% portions of the set are only smurf (DoS) attacks.

Table 5: Sample Distribution in the Data Sets

Dataset Name	Normal	DoS	Probe	U2R	R2L	Total
Whole KDD	19.8%	79.3%	0.84%	0.001%	0.02%	4,898,431
10% KDD	19.79%	79.2%	0.8%	0.01%	0.2%	494,020
Corrected Test	19.58%	73.9%	1.3%	0.02%	5.2%	311,029

The first two rows refer to the training sets. The third row refers to the testing set. The data is from [4].

2 For example, in the program used to generate the results of this thesis, the data structure used could only consume about 2GB in memory before throwing an exception.

Table 6: Redundant Records in the Training Set

	Original Records	**Distinct Records**	**% Redundant Records**
Attacks	3,925,650	262,178	93.32%
Normal	972,781	812,814	16.44%
Total	4,898,431	1,074,992	78.05%

The data is from [5].

Table 7: Redundant Records in the Testing Set

	Original Records	**Distinct Records**	**% Redundant Records**
Attacks	250,436	29,378	88.26%
Normal	60,591	47,911	20.92%
Total	311,027	77,289	75.15%

The data is from [5].

Bagheri, et al., in [5] were able to achieve a 98% classification rate, defined as the amount of correct results divided by the total results, using simple methods. Even when using the testing set, they achieved a minimum rate of 86%. In their initial results, they kept a counter for each record that was a count of how many of their learners correctly classified the record. Since they had seven different algorithms, each of which had three copies trained on different data sets, they had a total of 21 classifiers which gave the mentioned results and each record had a score from 0 to 21.

To more thoroughly test their algorithms, Bagheri, et al., in [5] created a new data set, called NSL-KDD. To create this set, they first removed all redundant records in the training and testing sets. From this reduced set, records were randomly sampled such that the difficulty was much harder. They achieved this by inversely weighting the amount of each group by the score counter, so that the number of records that were in the range of 0-

5 constituted 99.96% of the NSL-KDD set since they were 0.04% of the original training set. This resulted in a training set of 125,973 records and a testing set of 22,544 records. This data set has two implications. The first is that due to the inversely proportional difficulty the classification rates of various learners will vary by a larger amount, which makes analysis more reflective of actual performance. Secondly, since both the training and testing sets are reasonably sized, it is feasible to run experiments on the entire set without random sampling and comparisons between different researchers will be consistent. Details about the distributions of the NSL-KDD training and testing sets are shown in Tables 8 and 9.

Table 8: Randomly Selected Records from Training Data

Counter Value	Distinct Records	Percentage of Data	Selected Records
0-5	407	0.04%	407
6-10	768	0.07%	767
11-15	6,525	0.61%	6,485
16-20	58,995	5.49%	55,757
21	1,008,297	93.80%	62,557
Total	1,074,992	100.00%	125,973

To make the NSL-KDD training set, they removed duplicates from the full KDD training set and then randomly selected records inversely proportional to their difficulty [5].

Table 9: Randomly Selected Records from Testing Data

Counter Value	Distinct Records	Percentage of Data	Selected Records
0-5	589	0.76%	585
6-10	847	1.10%	838
11-15	3,540	4.58%	3,378
16-20	7,845	10.15%	7,049
21	64,468	83.41%	10,694
Total	77,289	100.00%	22,544

To make the NSL-KDD testing set, they removed duplicates from the full KDD testing set and then randomly selected records inversely proportional to their difficulty [5].

Out of the four attack types, R2L attacks are the most diverse in terms of execution and implementation. Each type of attack varies greatly in its signature and the target of the attack. Heuristic rules are often used to detect these attacks, which are generated by decision trees such as C4.5 and also by signature analysis. Decision tree rules are excellent when applied to the data they were generated from, they are hard to use and much less accurate if new records or features are added. Hence, it is hard to estimate their performance in novel situations and with unknown attacks. Because of the way a decision tree is generated, they make a best fit for the data set they are trained on; if novel data is introduced that does not fit within known patterns or if new features are added, a decision tree will not perform well by its very nature [6].

Rules generated by signature analysis are in general more comprehensible than those of a decision tree, may be better generalized, and are much easier to update. A challenge for signatures though is to determine proper thresholds for certain features, which if not done properly results in an ineffective rule. These can be determined by using C4.5 or other decision tree algorithms. The advantages to using any sort of heuristic rule is they require very little processing power and so can be run in real time and are very precise once defined well. The problem is that expert knowledge, which is not always easily distilled into if-then rules with precise thresholds, is required to perform signature analysis. Another challenge is that of generalizing the rules to unknown situations where noise is unpredictable. The expert system EMERALD was only able to detect 35% of R2L attacks from the DARPA testing set, and the winner of the KDD Cup '99 only detected 7.82% of the R2L attacks in the testing set [6].

Maheshkumar Sabhnani and Gursel Serpen in [6] analyzed both the KDD training and testing sets and came up with heuristic rules for two particular types of R2L attacks, Warezmaster and Warezclient. The Warezmaster attack exploits the guest account feature of an FTP server. Guests are normally not given write permissions, but if the administrator has given all users write permissions on the server guests can freely log in and upload files. In this particular case, the attacker will log in to the server and create a hidden directory and upload warez (illegally copied software) for others to later download. Relevant features in the data set are that an FTP session is in progress, files are being uploaded (large amounts of data coming from the source compared to the destination), hidden directories are made, and the user is a guest account. Two separate rules were developed, shown in Figures 2 and 3, and if either of them are true then a Warezmaster attack can be concluded. Using these rules, the authors were able to detect 65% of Warezmaster attacks with a low false positive rate of 0.005%. The authors hypothesize that the missed records were labeled incorrectly, as in these records either there was no data transferred from the source or no user was logged in. Since a Warezmaster attack requires both of these things to be true, this is likely a mislabeling.

$$\begin{gathered} \textit{If } (\textit{duration is long}) \wedge (\textit{the protocol is TCP}) \wedge (\textit{the service is FTP} \vee \textit{FTP Data}) \wedge \\ (\textit{a large amount of data is downloaded}) \wedge (\textit{no data is uploaded}) \\ \textit{Conclude a Warezmaster attack} \end{gathered}$$

Figure 2: Warezmaster Rule 2.1a from [6]: “If during an FTP session, large amount is data of sent from source as compared to destination then Warezmaster attack can be concluded.”

$$\begin{gathered} \textit{If } (\textit{the protocol is TCP}) \wedge (\textit{the service is FTP or FTP Data}) \wedge \\ (\textit{The hot indicator is small}) \wedge (\textit{The user is a guest}) \\ \textit{Conclude a Warezmaster attack} \end{gathered}$$

Figure 3: Warezmaster Rule 2.1b from [6]: “If a guest has logged in through an FTP connection, and hidden directories are created then Warezmaster attack can be concluded.”

The Warezclient attack is the other half of the Warezmaster attack; in this attack a user downloads the warez. This attack is harder to detect, as in general it mirrors the normal download process from an FTP server. The only feature that can be utilized is that of downloading files from either hidden directories or directories that guest users should not be able to access. The hot feature encapsulates this, and if many hot indicators are noticed in a small amount of time then a Warezclient attack is probably taking place. The results for the Warezclient rules were not as successful as Warezclient, with only a 30% detection rate (though with 0 false alarms). This is because downloading illegal files is very similar to downloading legal files. Also, there were some records that had a high enough hot value (and therefore accessing hidden directories or directories a guest should not be accessing) to be considered an attack but didn't have any file downloads occur. This is most likely because the user accessed a hidden directory, whether legal or not, but didn't download any of the files in it; hence it is not really a Warezclient attack. The combined detection rate was 53.08%, with 70 false alarms out of 1,383,435 non-Warez records. The rule is shown in Figure 4.

$$\textit{If}\ (\textit{the duration is small}) \wedge (\textit{the protocol is TCP}) \wedge (\textit{the service is FTP or FTP Data}) \wedge$$
$$(\textit{the user is logged in or logged in as a guest}) \wedge (\textit{the amount of hot indicators is large})$$
$$\textit{Conclude a Warezclient attack}$$

Figure 4: Warezclient Rule 2.2 from [6]: “If a user, during an FTP session, triggers notably many hot indicators to be set in a small duration of time then the user maybe downloading illegally posted software from the server.”

CHAPTER 2

METHODOLOGY

Based on the research detailed previously, the NSL-KDD data sets were used as a comparison to the original set, as they provided a more complete test for my learners and were much easier to computationally handle. Before doing any learning on the data though, it needed to be preprocessed. The flag, label, service, and protocol features were strings of varied ranges, so they were mapped to integer values and then linearly scaled from [0, 1]. 15 other features needed to be linearly scaled to [0, 1], as they had max ranges from 3 to 58,329. Two features, source bytes and destination bytes, ranged from [0, 1.3 billion] and so were logarithmically scaled (base 10) to [0, 9.14] [7]. Once the data was preprocessed, it was run though the K-Nearest Neighbor (KNN) algorithm, which was enhanced with a comparison of the guessed result and the result from the signatures developed by [6]. In addition, the data was tested on the C5.0 decision tree algorithm developed by [8] to compare against the results of the third place contestant, who had used C4.5. The use of Weka was considered to supplement results as well, but after initial use the numbers did not substantially differ enough to justify further explorations.

For the decision tree, one run was done with the training set being the 10% partition of the KDD data set given on [1], referred to in the result section as "the KDD10% tree" and the other run using the NSL-KDD training set, referred to as "the

NSL-KDD tree”. Both runs used the NSL-KDD data set as the testing set. A third run was done using the KDD10% training and testing sets.

The first KNN run was done with K = 1 and with the KDD10% training and testing sets to give a comparison with the KDD Cup 10th place participant. The NSL-KDD testing set was swapped in and KNN was run again. Keeping the NSL-KDD testing set, the NSL-KDD training set was swapped in and KNN was run again. From this point, KNN was repeated from K = 2 to K = 10, using the NSL-KDD training and testing sets each time as results were negligibly different from using the KDD10% training set and the NSL-KDD testing set.

CHAPTER 3

RESULTS

Overall, the first two decision trees had about the same accuracy, with the KDD10% tree having a 2% higher value. The KDD10% tree had a much lower overall average cost, 0.5515 compared to 0.6369. Both trees had a worse result than just classifying all test examples as DoS, though the tress were using the NSL-KDD training set instead of the KDD set. The recall for the KDD10% tree was better too, with R2L being almost 16% higher and all others having a difference of < 2%. For definitions of accuracy, recall, and precision see Table 10. While the KDD10% tree also did better on precision for normal, probing, and DoS attacks, the NSL-KDD tree did 17% better on U2R attacks and 20% better on R2L attacks. The results are shown in detail in Figure 5. This shows that the NSL-KDD set is more diverse and hence harder to correctly classify, and also that it contains a larger percent of the rare U2R and R2L attacks. The third decision tree did surprisingly well. It had an average cost of 0.2291 and an accuracy of 92.70%. The accuracy was only 0.01% worse than the first place contestant, and had a lower average cost by 0.004. It also had a higher probing precision and a slightly higher R2L recall, but at the cost of much lower precisions for U2R and R2L.

Decision Tree 1	**KDD10%**		**NSL-KDD**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9440	191	70	5	5	97.21%
	1	668	1600	123	0	30	66.09%
(actual)	**2**	1329	44	5914	0	171	79.30%
	3	451	58	3	14	7	2.63%
	4	1561	127	31	8	694	28.67%
(Precision)		70.19%	79.21%	96.30%	51.85%	76.52%	
Total:		22544					
Error:		4882					
Amount Correct:		17662					
Accuracy:		78.34%					
Average Cost:		0.551543648					

Decision Tree 2	**NSL-KDD**		**NSL-KDD**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9421	204	84	1	1	97.01%
	1	610	1553	249	0	9	64.15%
(actual)	**2**	1463	84	5911	0	0	79.26%
	3	358	164	1	9	1	1.69%
	4	2107	3	0	3	308	12.72%
(Precision)		67.49%	77.34%	94.65%	69.23%	96.55%	
Total:		22544					
Error:		5342					
Amount Correct:		17202					
Accuracy:		76.30%					
Average Cost:		0.6368878637					

Decision Tree 3	**KDD10% Training**		**KDD10% Test**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	60283	226	73	6	5	99.49%
	1	704	3288	144	0	30	78.92%
(actual)	**2**	6137	48	223492	0	176	97.23%
	3	2535	76	3	14	8	0.53%
	4	12140	342	43	8	1248	9.06%
(Precision)		73.70%	82.61%	99.88%	50.00%	85.07%	
Total:		311029					
Error:		22704					
Amount Correct:		288325					
Accuracy:		92.70%					
Average Cost:		0.2290622418					

Figure 5: Results of the decision tree on various data sets.

Table 10: Definitions

Name	Formula	Definition
Accuracy	(True Positive + False Negative) / (Total)	The percentage of correctly identified records
Recall	(True Positive) / (True Positive + False Negative)	The percentage of data that was labeled correctly
Precision	(True Positive) / (True Positive + False Positive)	The percentage of data with a given classification label that was actually that label.

Definitions of terms used for the analysis of results.

Results using KNN without using signature rules closely matched the results of the tenth place contestant for the most part. Accuracy was insignificantly higher, average classification cost was better by 0.008, recall for probing was better by 5%, their recall rates for U2R and R2L were better by 2.5% and 0.17%, and their precision rates for probing, U2R, and R2L were better by 3.82%, almost 80%, and almost 14% respectively. If different partitions of the training data were used, these differences are easily explainable. With signature rules enabled KNN classified 1031 more R2L attacks, of which only 14 were false alarms. This resulted in a very high precision rate of 97.29% for R2L, as well as an increase in precision for normal by 0.90%. The recall for R2L increased by 6.8% at the cost of a 0.02% decrease in recall for normal. These changes caused an overall accuracy of 92.67% and an average cost of 0.2313, which actually beats the first place winner of the KDD Cup by 0.0017.

The results from using the NSL-KDD testing set only were fairly similar to the decision trees, with the run without signatures having a cost of 0.6333 and the run with signatures having a cost of 0.5148. With and without signatures, the results from using both the NSL-KDD training and testing set were different from using only the NSL-KDD testing set by about 0.005 in cost and 0.34% in accuracy. Averaged out between the 10 tests, KNN with signatures had an average cost of 0.5338, accuracy of 79.37%, normal precision of 71.40%, normal recall of 96.72%, R2L precision of 97.28%, and a R2L recall of 30.02%. KNN without the signatures had an average cost of 0.6540, accuracy of 76.39%, normal precision of 67.90%, normal recall of 96.84%, R2L precision of 87.10%, and a R2L recall of 1.80%. All other values stayed the same, since only R2L values were effected by the rules and according to [6] all the Warez attacks being misclassified were being classified as normal. This shows that, while having a minimal effect on everything else, signatures can greatly increase the effectiveness of a learner on a particular type of attack. Warez attacks were about 30% of the R2L attacks in the testing set, matching the 30% average recall rate. The results for K = 1 were slightly above average. The lowest average cost though both with and without signatures was for K = 8, without signatures it was 0.5928 and with signatures it was 0.4769. Results of these tests are shown in Figures 6 through 15.

KNN Results 1.1		**KDD10% Training**	**KDD10% Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	60274	239	70	9	1	99.47%
	1	617	3350	196	0	3	80.41%
(actual)	**2**	5850	483	223520	0	0	97.24%
	3	2490	110	0	24	12	0.91%
	4	13618	3	2	100	58	0.42%
(Precision)		72.75%	80.05%	99.88%	18.05%	78.38%	
		Total:	311029				
K = 1		Error:	23803				
No Rules		Amount Correct:	287226				
		Accuracy:	92.35%				
		Average Cost:	0.2443276993				

KNN Results 2.1		**KDD10% Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9436	203	66	5	1	97.17%
	1	594	1648	176	0	3	68.07%
(actual)	**2**	1055	179	6224	0	0	83.45%
	3	408	92	0	23	10	4.32%
	4	2263	3	2	100	53	2.19%
(Precision)		68.60%	77.55%	96.23%	17.97%	79.10%	
		Total:	22544				
K = 1		Error:	5160				
No Rules		Amount Correct:	17384				
		Accuracy:	77.11%				
		Average Cost:	0.6333392477				

KNN Results 3.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9413	232	62	3	1	96.93%
	1	453	1809	158	0	1	74.72%
(actual)	**2**	1170	119	6169	0	0	82.72%
	3	426	83	0	20	4	3.75%
	4	2186	91	0	95	49	2.02%
(Precision)		68.97%	77.51%	96.56%	16.95%	89.09%	
		Total:	22544				
K = 1		Error:	5084				
No Rules		Amount Correct:	17460				
		Accuracy:	77.45%				
		Average Cost:	0.6281937544				

Figure 6: KNN Results 1.1-3.1.

KNN Results 4.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9434	217	59	1	0	97.15%
	1	504	1746	170	0	1	72.12%
(actual)	**2**	1644	112	5702	0	0	76.45%
	3	432	83	0	14	4	2.63%
	4	2372	30	0	2	17	0.70%
(Precision)		65.58%	79.80%	96.14%	82.35%	77.27%	

	Total:	22544
K = 2	Error:	5631
No Rules	Amount Correct:	16913
	Accuracy:	75.02%
	Average Cost:	0.6922019163

KNN Results 5.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9438	214	58	1	0	97.19%
	1	529	1719	173	0	0	71.00%
(actual)	**2**	1644	107	5707	0	0	76.52%
	3	441	83	0	9	0	1.69%
	4	2396	7	0	2	16	0.66%
(Precision)		65.32%	80.70%	96.11%	75.00%	100.00%	

	Total:	22544
K = 3	Error:	5655
No Rules	Amount Correct:	16889
	Accuracy:	74.92%
	Average Cost:	0.6961053939

KNN Results 6.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9448	205	58	0	0	97.29%
	1	561	1688	172	0	0	69.72%
(actual)	**2**	1648	107	5703	0	0	76.47%
	3	443	83	0	7	0	1.31%
	4	2407	0	0	0	14	0.58%
(Precision)		65.13%	81.04%	96.12%	100.00%	100.00%	

	Total:	22544
K = 4	Error:	5684
No Rules	Amount Correct:	16860
	Accuracy:	74.79%
	Average Cost:	0.6987224982

Figure 7: KNN Results 4.1-6.1.

KNN Results 7.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9452	201	58	0	0	97.33%
	1	669	1581	171	0	0	65.30%
(actual)	**2**	1648	106	5704	0	0	76.48%
	3	457	72	0	7	0	1.31%
	4	2407	0	0	0	14	0.58%
(Precision)		64.59%	80.66%	96.14%	100.00%	100.00%	
		Total:	22547				
K = 5		Error:	5789				
No Rules		Amount Correct:	16758				
		Accuracy:	74.32%				
		Average Cost:	0.703996097				

KNN Results 8.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9364	272	68	6	1	96.43%
	1	289	1945	186	0	1	80.34%
(actual)	**2**	946	226	6286	0	0	84.29%
	3	400	105	0	21	7	3.94%
	4	2225	108	2	58	28	1.16%
(Precision)		70.81%	73.23%	96.09%	24.71%	75.68%	
		Total:	22544				
K = 6		Error:	4900				
No Rules		Amount Correct:	17644				
		Accuracy:	78.26%				
		Average Cost:	0.614930802				

KNN Results 9.1		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9341	289	69	7	5	96.19%
	1	253	1980	187	0	1	81.78%
(actual)	**2**	908	247	6303	0	0	84.51%
	3	384	107	0	31	11	5.82%
	4	2124	120	2	106	69	2.85%
(Precision)		71.80%	72.18%	96.07%	21.53%	80.23%	
		Total:	22544				
K = 7		Error:	4820				
No Rules		Amount Correct:	17724				
		Accuracy:	78.62%				
		Average Cost:	0.5980748758				

Figure 8: KNN Results 7.1-9.1.

KNN Results 10.1		**NSL-KDD Training**	**NSL-KDD Test**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9336	296	66	8	5	96.14%
	1	230	2025	165	0	1	83.64%
(actual)	**2**	891	264	6303	0	0	84.51%
	3	381	110	0	32	10	6.00%
	4	2110	124	2	113	72	2.97%
(Precision)		72.10%	71.83%	96.44%	20.92%	81.82%	
		Total:	22544				
K = 8		Error:	4776				
No Rules		Amount Correct:	17768				
		Accuracy:	78.81%				
		Average Cost:	0.5927519517				

KNN Results 11.1		**NSL-KDD Training**	**NSL-KDD Test**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9411	226	61	8	5	96.91%
	1	492	1751	176	0	2	72.33%
(actual)	**2**	1456	168	5833	1	0	78.21%
	3	416	75	0	35	7	6.57%
	4	2214	18	1	113	75	3.10%
(Precision)		67.27%	78.24%	96.08%	22.29%	84.27%	
		Total:	22544				
K = 9		Error:	5439				
No Rules		Amount Correct:	17105				
		Accuracy:	75.87%				
		Average Cost:	0.6580908446				

KNN Results 12.1		**NSL-KDD Training**	**NSL-KDD Test**	**(classified as)**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9403	231	61	9	7	96.83%
	1	461	1780	176	0	4	73.52%
(actual)	**2**	1484	172	5801	1	0	77.78%
	3	407	84	0	36	6	6.75%
	4	2190	27	2	121	81	3.35%
(Precision)		67.43%	77.59%	96.04%	21.56%	82.65%	
		Total:	22544				
K = 10		Error:	5443				
No Rules		Amount Correct:	17101				
		Accuracy:	75.86%				
		Average Cost:	0.6568931867				

Figure 9: KNN Results 10.1-12.1.

NSL-KDD Training	**NSL-KDD Test**		**Average of Results**				
		0	**1**	**2**	**3**	**4**	(Recall)
	0						96.84%
	1						74.45%
(actual)	**2**						79.79%
	3						3.98%
	4						1.80%
(Precision)		67.90%	77.28%	96.18%	48.53%	87.10%	

K = 1 to 10	Accuracy:	76.39%
	Average Cost:	0.65399613

Figure 10: Average of KNN Results 3.1-12.1.

KNN Results 1.2		KDD10% Training	KDD10% Test	(classified as)			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	60260	239	70	9	15	99.45%
	1	617	3350	196	0	3	80.41%
(actual)	**2**	5850	483	223520	0	0	97.24%
	3	2490	110	0	24	12	0.91%
	4	12601	3	2	100	1075	7.80%
(Precision)		73.65%	80.05%	99.88%	18.05%	97.29%	

	Total:	311029
K = 1	Error:	22800
Rules	Amount Correct:	288229
	Accuracy:	92.67%
	Average Cost:	0.2313385569

KNN Results 2.2		KDD10% Training	NSL-KDD Test	(classified as)			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9424	203	66	5	13	97.04%
	1	594	1648	176	0	3	68.07%
(actual)	**2**	1055	179	6224	0	0	83.45%
	3	408	92	0	23	10	4.32%
	4	1589	3	2	100	727	30.03%
(Precision)		72.10%	77.55%	96.23%	17.97%	96.55%	

	Total:	22544
K = 1	Error:	4498
Rules	Amount Correct:	18046
	Accuracy:	80.05%
	Average Cost:	0.514815472

KNN Results 3.2		NSL-KDD Training	NSL-KDD Test	(classified as)			
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9401	232	62	3	13	96.81%
	1	453	1809	158	0	1	74.72%
(actual)	**2**	1170	119	6169	0	0	82.72%
	3	426	83	0	20	4	3.75%
	4	1511	91	0	95	724	29.90%
(Precision)		72.53%	77.51%	96.56%	16.95%	97.57%	

	Total:	22544
K = 1	Error:	4421
Rules	Amount Correct:	18123
	Accuracy:	80.39%
	Average Cost:	0.5094925479

Figure 11: KNN Results 1.2-3.2.

KNN Results 4.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9422	217	59	1	12	97.02%
	1	504	1746	170	0	1	72.12%
(actual)	**2**	1644	112	5702	0	0	76.45%
	3	432	83	0	14	4	2.63%
	4	1665	30	0	2	724	29.90%
(Precision)		68.94%	79.80%	96.14%	82.35%	97.71%	

	Total:	22544
K = 2	Error:	4936
Rules	Amount Correct:	17608
	Accuracy:	78.11%
	Average Cost:	0.5678229241

KNN Results 5.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9426	214	58	1	12	97.07%
	1	529	1719	173	0	0	71.00%
(actual)	**2**	1644	107	5707	0	0	76.52%
	3	441	83	0	9	0	1.69%
	4	1688	7	0	2	724	29.90%
(Precision)		68.66%	80.70%	96.11%	75.00%	98.37%	

	Total:	22544
K = 3	Error:	4959
Rules	Amount Correct:	17585
	Accuracy:	78.00%
	Average Cost:	0.5715489709

KNN Results 6.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9436	205	58	0	12	97.17%
	1	561	1688	172	0	0	69.72%
(actual)	**2**	1648	107	5703	0	0	76.47%
	3	443	83	0	7	0	1.31%
	4	1698	0	0	0	723	29.86%
(Precision)		68.45%	81.04%	96.12%	100.00%	98.37%	

	Total:	22544
K = 4	Error:	4987
Rules	Amount Correct:	17557
	Accuracy:	77.88%
	Average Cost:	0.5739886444

Figure 12: KNN Results 4.2-6.2.

KNN Results 7.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9440	201	58	0	12	97.21%
	1	669	1581	171	0	0	65.30%
(actual)	**2**	1648	106	5704	0	0	76.48%
	3	457	72	0	4	0	0.75%
	4	1698	0	0	0	723	29.86%
(Precision)		67.86%	80.66%	96.14%	100.00%	98.37%	
		Total:	22544				
K = 5		Error:	5092				
Rules		Amount Correct:	17452				
		Accuracy:	77.41%				
		Average Cost:	0.5793559262				

KNN Results 8.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9352	272	68	6	13	96.30%
	1	289	1945	186	0	1	80.34%
(actual)	**2**	946	226	6286	0	0	84.29%
	3	400	105	0	21	7	3.94%
	4	1528	108	2	58	725	29.95%
(Precision)		74.73%	73.23%	96.09%	24.71%	97.18%	
		Total:	22544				
K = 6		Error:	4215				
Rules		Amount Correct:	18329				
		Accuracy:	81.30%				
		Average Cost:	0.4923261178				

KNN Results 9.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9329	289	69	7	17	96.07%
	1	253	1980	187	0	1	81.78%
(actual)	**2**	908	247	6303	0	0	84.51%
	3	384	107	0	31	11	5.82%
	4	1463	120	2	106	730	30.15%
(Precision)		75.62%	72.18%	96.07%	21.53%	96.18%	
		Total:	22544				
K = 7		Error:	4171				
Rules		Amount Correct:	18373				
		Accuracy:	81.50%				
		Average Cost:	0.4818577005				

Figure 13: KNN Results 7.2-9.2.

KNN Results 10.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9324	296	66	8	17	96.01%
	1	230	2025	165	0	1	83.64%
(actual)	**2**	891	264	6303	0	0	84.51%
	3	381	110	0	32	10	6.00%
	4	1451	124	2	113	731	30.19%
(Precision)		75.95%	71.83%	96.44%	20.92%	96.31%	
		Total:	22544				
K = 8		Error:	4129				
Rules		Amount Correct:	18415				
		Accuracy:	81.68%				
		Average Cost:	0.476889638				

KNN Results 11.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9399	226	61	8	17	96.79%
	1	492	1751	176	0	2	72.33%
(actual)	**2**	1456	168	5833	1	0	78.21%
	3	416	75	0	35	7	6.57%
	4	1558	18	1	113	731	30.19%
(Precision)		70.56%	78.24%	96.08%	22.29%	96.57%	
		Total:	22544				
K = 9		Error:	4795				
Rules		Amount Correct:	17749				
		Accuracy:	78.73%				
		Average Cost:	0.5427608233				

KNN Results 12.2		**NSL-KDD Training**	**NSL-KDD Test**		**(classified as)**		
		0	**1**	**2**	**3**	**4**	(Recall)
	0	9391	231	61	9	19	96.70%
	1	461	1780	176	0	4	73.52%
(actual)	**2**	1484	172	5801	1	0	77.78%
	3	407	84	0	36	6	6.75%
	4	1539	27	2	121	732	30.24%
(Precision)		70.70%	77.59%	96.04%	21.56%	96.19%	
		Total:	22544				
K = 10		Error:	4804				
Rules		Amount Correct:	17740				
		Accuracy:	78.69%				
		Average Cost:	0.5424503194				

Figure 14: KNN Results 10.2-12.2.

NSL-KDD Training	**NSL-KDD Test**		**Average of Results**			
		0	**1**	**2**	**3**	**4**	(Recall)
	0						96.72%
	1						74.45%
(actual)	**2**						79.79%
	3						3.92%
	4						30.02%
(Precision)		71.40%	77.28%	96.18%	48.53%	97.28%	

K = 1 to 10	Accuracy:	79.37%
	Average Cost:	0.53384936

Figure 15: Average of KNN Results 3.2-12.2.

CHAPTER 4

CONCLUSION

Even though R2L attacks are a comparatively small amount of the total data sets (10.74% of NSL-KDD Testing, 5.20% of the KDD testing), misclassifying them carries a high cost since they are dangerous attacks. Hence, anything that greatly improves the detection of these types of attacks will result in a learner that has a significantly decreased average classifying cost. Using simple methods, in this case KNN and C5.0 decision trees, gave impressive results. With the original data sets, the results were on par with the first place winner once signatures for Warezmaster and Warezclient attacks were implemented. While using the NSL-KDD data sets, implementing signatures decreased the average cost by 0.1201 and increased the recall of R2L by 28.12%. There are many more signatures to discover for the various U2R and R2L attacks, each of which would measurably improve the capabilities of a learner on even the NSL-KDD data set.

REFERENCES

[1] KDD Cup 1999: Tasks. Retrieved February 12, 2013 from http://www.sigkdd.org/kddcup/index.php?section=1999&method=task.

[2] Miheev, V., Shabalin, I., & Vopilov, A. (2000). The MP13 Approach to the KDD'99 Classifier Learning Contest. *SIGKDD Explorations*, *1(2)*, 76-77.

[3] Elkan, C. (2000). Results of the KDD’99 Classifier Learning. *SIGKDD Explorations, 1(2)*, 63-64.

[4] Jazzar, M., Jazzer, M., & Jantan, A. (2010). On Multi-Classifier Systems for Network Anomaly Detection and Features Selection. *International Journal of Computer Science and Information Security, 8(2)*, 254-263.

[5] Bagheri, E., Lui, W., Ghorbani, A., & Tavallaee, M. (2009). A Detailed Analysis of the KDD CUP 99 Data Set. *In Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications,* 53-58.

[6] Sabhnani, M., & Serpen, G. (2003). KDD Feature Set Complaint Heuristic Rules for R2L Attack Detection. *Proceedings of the International Conference on Security and Management 2003*, 310-316.

[7] Bahrololum, M., Khaleghi, M., & Salahi, E. (2009). Anomaly Intrusion Detection Design using Hybrid of Unsupervised and Supervised Neural Network. *International Journal of Computer Networks & Communications, 1(2),* 26-33.