Evaluation of data mining techniques for malware detection using system-call sequences

Amirreza Soudi
Concordia University
am_soudi@ece.concordia.ca

Shayan Eskandari
Concordia University
e_eskand@ece.concordia.ca

ABSTRACT
In this paper we want to use various modeling techniques of data mining on our dataset. The purpose of this paper is to evaluate each model and also to find out impact of sequence window size on detection of anomalies. The expected result is to have a good window size and the best model to detect anomalies. The data has been gathers from Anubis dataset that contains normal and malware traces extracted from Anubis application.

Keywords
System-call sequence, data-mining, anomaly detection

1. INTRODUCTION
Sequence data are found in a wide variety of application domains such as intrusion detection, bioinformatics, weather prediction, system health management, etc. Anomaly detection for sequence data is an important topic of research as it often leads to detection of critical information such as intrusions, faults, and system failures [9].

In host-based intrusion detection system, signature-based detection approaches focus on detecting attack patterns while anomaly detection system (ADS) approaches by modeling the normal system behavior and flag any deviation as anomaly. Host-Based IDSs (HIDS) are designed to monitor the host system activities and state. There are lots of works done related to this field. Our approach is based on system call sequences, with this assumption that any anomaly would have an impact in this level; cause system calls are the gateway between user and kernel mode in the operating system.

One of the biggest problem of current host based intrusion detection system (HIDS) is the high number of false positive alarms they produce, to have an operational HIDS we should decrease false positive rate as much as possible. Therefor proposing new techniques with the same accuracy and less false alarms are a necessity of today security world.

Table 1 Comparison between Contemporary IDS algorithms [1]

Algorithm	Detection Rate [%]	False Alarm Rate [%]
Data mining of audit files [60]	80.2	Not cited
Multivariate statistical analysis of audit data [33]	90	40
HMM and entropy analysis of system calls [61]	91.7	10.0
System call n-gram sliding window (assorted decision engines) [46]	$95.3 < DR < 96.9$	~ 6.0
RBF ANN analysing system calls [31]	96 mean	5.4 mean
MLP ANN on subset of KDD98 [62]	99.2	4.94
SVM on subset of KDD98 [62]	99.6	4.17
kNN with Smooth Binary Weighted RBF [63]	96.3	6.2
Rough Set Clustering [64]	95.9	7.2

In table [1] the comparison of many techniques and their relative detection rate and false alarm rate are shown. The false alarm rates for each of them are not good enough so we should try to improve it yet we should keep or increase detection rate.

These days the demand for accurate with low false positive HIDS is increasing; to respond this demand researcher developed many technique, some proposed methods used the help of data mining methods. By using data mining technique we can apply 2 different ways for detecting anomalous sequence data. First one is to train our model just based on normal data gathering from clean system (one-class classifier model) and after that apply our new model on test date (real data) to label them as normal or abnormal. Other approach is to train our model based on both normal and abnormal sequence (two-class classifier) getting from infected system and apply our new model on real data to label them. One of the difficulty of second approach is how we can generate dataset of normal and abnormal data, how we can label them correctly?

In our project we have some challenges, first of all we should some how gather the desired information from the data set, which contains normal and anomalous traces. We solve this problem by using sequence time-delay embedding (STIDE) technique, we have access to sequence of normal data so we would make different window sized subsets of this sequence (by size 5 to 8) and store them as our normal dataset and did the same with malware sequence. After that we tried to find each sequence in malware sequence file inside the normal dataset, if we find that window we label it as normal otherwise as an anomaly. Second of all, since we have huge amount of malware traces we should find a way to pick very little but comprehensive sort of malwares, otherwise the learning model and evaluating the testing data would be time-consuming. For solving this problem we try to select our malware data randomly through all part of the malware data set.

2. Related Work
So many different architectures exist for IDS. IDS can be centralized or distributed across many machines. Most of all IDS are centralized. Furthermore, another categorization of IDS is to be host-based or network-based; the former type monitors activity on a single computer, whereas the latter type monitors activity over a network. Network-based IDS can monitor information collected from audit logs from many different hosts (multi-host monitoring) or they can monitor network traffic. Regardless of other architectural considerations, any IDS must have three components: Data collection (and reduction), data classification and data reporting [3].

In general there are two basic approaches to intrusion detection, misuse intrusion detection and anomaly intrusion detection. In misuse intrusion detection, intrusion signatures are used to try to identify intrusions when they happen. In anomaly intrusion detection, it is assumed that the nature of the intrusion is unknown, but that the intrusion will result in behavior different from that normally seen in the system. Most previous work on anomaly intrusion detection has determined profiles for user behavior. Intrusions are detected when a user behaves out of character [2].

There are so many works on anomaly detection, the very first idea was for Forrest [2], she tried to create a series of window size of 8 from sequence of traces (just system call numbers) and create a normal model based on these files. There are two stages to the proposed algorithm. In the first stage, they scan traces of normal behavior and build up a database of characteristic normal patterns (observed sequences of system calls). In the second stage, they scan new traces that might contain abnormal behavior, looking for patterns not present in the normal database. They named this technique STIDE. After that Wagner showed that by using window size of 6 we can get better results [10].

The analysis of system call patterns is usually performed without considering the arguments passed to each system call. This process loses some information, but still allows for accurate classifications based on the system call sequences. Researchers such as Wagner [7] try to add argument while creating a normal model; they claim it is easy for attacker to evade IDS when it is just system call sequence being used, so they suggest that we should use extra information for creating a model such as arguments.

Besides above traditional approach some researcher try to solve these problem by using sequential pattern modeling [8] instead of traditional machine learning algorithm they tried Markov Model, Hidden Markov Model [8], Extreme Machine learning (EML) [1] to build their model based on sequence of traces. The ELM methodology is an extremely powerful decision engine in the artificial neural network family. Like many types of decision engine, neural networks suffer from a high training overhead, traditionally requiring huge amount of resources and many iterations to reach an acceptable level of training.

The key characteristic of an ELM is that it completes the training phase in one pass, avoiding much of the traditional training problem related to neural networks. The training speed comes with a high processing overhead, but the impact of this requirement is small, as HIDS have access to the CPU of the host itself. Alternatively, offline training is possible as well, allowing for a large proportion of the processing load to be completed prior to end-user deployment [1].

As you can see in Figure 1 there are 3 different approaches for finding abnormal behavior in system call traces.

3. Problem statement

The major advantage of ADS is its ability to detect novel attacks, however it will generate large number of false alarms. Typically, false alarms are due to poor modeling, invalid assumptions, and unrepresentative dataset for training and testing.
Any deviation in syscall sequences would be flagged as anomaly, since we have trained and stored syscall sequences of normal behavior of the system. Our comparison approach has variables defined as sequence window size, the number of shifts for the next window and frequency of each system call sequence.
The result of this comparison would help us to evaluate the existing solutions with different window sizes, in order to find the proper method or a combination of methods that have the minimum false alarms.
Most of the datasets used in related works are relatively old and the need for new representative dataset to evaluate the new techniques is increasing. So we pick new set of data set, explained later, for our work. With testing new malware traces on our model we would find out what type of data mining algorithms would work with high accuracy and low false positive.

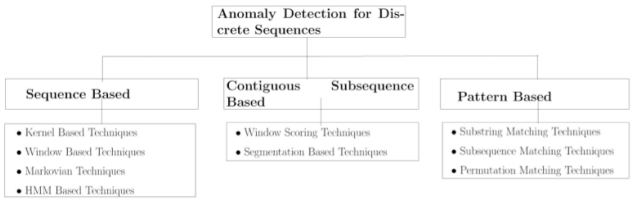


Figure 1 Taxonomy of existing technique

For finding more accurate solution with less false positive we compare many different modeling algorithms. The model we used is support vector machine (SVM), random forest, linear model, boost and neural network.

- SVM [4]: it is a supervised learning algorithm used for categorization of data and classification. Usually SVM work on data with two different classes. The algorithms try to solve an equation to find the best line to separate two groups of data. In modern model we can find non-linear result.

- Random Forrest [5]: it is kind of multiple learning model which is used different decision tree to train their model and try to output the class that is the mode of the classes output by individual trees. In simple word it uses voting algorithm by giving the input to many various decision tree and put the data to class with highest number of vote.

- Linear Model: It is a SVM ancestor, it try to divide a data to two different groups by use an object's characteristics.

- Boost [6]: In face boosting is not a specific model it is concept that can be used by many model. Boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. It is kind of weighted algorithm and during each iteration it tries to reweight data and calculate the result again.

- Neural Network: It is a adaptive learning model, so it changes its structure during learning phase. This model usually used for modeling complex. The first version of this algorithm was linear but the modern ones are non-linear statistical data model. This model usually

used when each class member has different type of parameters and use weight for each node or parameters.

4. Proposed Method

4.1 Anubis Dataset

The dataset we started working on is called Anubis[1] dataset. Anubis is a tool for analyzing malware. To this end, the binary executable is run in an emulated environment and its actions are monitored. They used four different datasets in their experiments. The first is a collection of execution traces of more than a thousand malware samples randomly extracted from Anubis. The second dataset contains 180 GB of execution traces collected from 10 different real-world machines, where they observed normal day-to-day operation of regular computer users. The third dataset (called Anubis-good) contains the traces of 36 normal application executed under Anubis.

Each part contains of related system calls and information regarding the return value and the file, which called the system call. For our purpose we had to parse the dataset to extract just the proportion of the dataset that we need to use. This would be the sequence of system calls.

4.1.1 Data-Cleaning

We used a scripting language, Python, to write a parser to extract the desired information (system call id sequence) from the dataset. Also there were some mistakes in the syntax of the dataset, which made some false statements the preliminary results.

4.1.2 Sequence Window Size

One measurement that we want to discuss is the sequence window size. That is the number of system calls in each sequence of size W. As one of the measurements that we want to discuss is the window size, we wrote another python code to extract desired window sequence from the result of previous parser, we ran our program on the dataset to extract the sequences of 5,6,7 and 8 system calls in each window.

4.2 Our Methodology

The objective of this project is to evaluation the performance of several data mining techniques such as Support Vector Machine (SVM), Markov Model, Sequence Time-Delay Embedding (Stide) on a newly acquired syscall dataset.

This involves preprocessing (cleaning, partitioning and labeling) of the dataset as well as varying and optimizing model's parameters in order to evaluate the accuracy and efficiency of each technique.

For the preprocessing purpose we parse the dataset to extract the desired information as the input for the methodologies being compared. This data would be the sequence of system call numbers, if we need more information such as process id, etc. we can extract them easily in the future. As for partitioning, we separated and stored the sequences in W-sized windows, as we change W to compare each method different window size to see affect of this change in the performance.

We want to apply different techniques of data mining on the dataset while modifying the variables like window size of the sequences. With having these results we could apply comparison methods (e. g. Receiver operating characteristics (ROC)) to see which one is more applicable to real time systems from the ratio of detection and trustworthiness of the alarms. Also this would result in having hybrid window size solutions for different situations.

As explained above we use 5 different modeling algorithms, we gave our dataset as an input for each of them by help of the R. after that we collect different result such as accuracy of predicting correct class, false positive, actual flag versus predicted flag. For each modeling algorithms we used different parameters to tune our model, for example for SVM we see from the result that our best option is to used Radial-Basis as a kernel.

5. Experiment Results

We used all the introduced models on the dataset with different sequence window sizes (5,6,7,8).

Table 2 Anticipated results of different classifier model

flag	boost	random fore	SVM	Linear	Neural	kmeans
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	0
1	1	1	0	0	0	0
0	0	0	0	0	0	0
1	1	1	1	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	1	0
1	0	1	0	0	0	0
0	0	0	0	0	0	1
1	0	1	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	1	1	1	0	1	1
0	0	0	0	0	0	1

To better understand the results we plotted the result of the testing and validating the data on receiver operating characteristic (ROC) curve. ROC is a graphical plot that shows the performance of a binary classifier. It is created by plotting the fraction of true positives out of the positives, true positive rate, versus the fraction of false positives out of the negatives, false positive rate. The result is shown at Appendix 1.

By just looking at the ROC curves it will show that Random forest and Boost have the overall best results. For a closer look we check the accuracy of each model in each window size. The result is on Table 1 to 4.

[1] http://anubis.iseclab.org/

Table 3 - Window size W = 5

Model	*Accuracy (%)*
Random Forest	98.79
Boost	91.07
SVM	77.71
Linear Model	65.93
Neural Network	73.62

Table 4 - Window size W = 6

Model	*Accuracy (%)*
Random Forest	98.43
Boost	89.78
SVM	81.36
Linear Model	67.21
Neural Network	50

Table 5 - Window size W = 7

Model	*Accuracy (%)*
Random Forest	98.38
Boost	88.06
SVM	84.79
Linear Model	66.41
Neural Network	50

Table 6 - Window size W = 8

Model	*Accuracy (%)*
Random Forest	98.33
Boost	86.79
SVM	85.92
Linear Model	67.86
Neural Network	68.98

Now that we have analyzed the result we can say that within the boundaries of this experiment window size of 5 has the best results so far. Now we have to check the error ratio to better understanding the results. We would just include the error matrix of window size of 5 for each model as this would be enough for the main purpose of this project.[Table 5,6,7,8]

Table 7 - Random Forrest Error Matrix

Actual/Predicted	0	1
0	80	1
1	5	15

Table 8 - Boost Error Matrix

Actual/Predicted	0	1
0	80	0
1	11	9

Table 9 - SVM Error Matrix

Actual/Predicted	0	1
0	80	0
1	12	8

Table 10 – Neural Network Error Matrix

Actual/Predicted	0	1
0	78	3
1	13	6

The overall error of Random forest would be 5.16 % that has the less value with in the other ones, so it would be considered as the best model in this case.

6. Conclusion

By studying the results we can see the on these tested models window size 5 has the best overall. Random Forest is the leading model on all window sizes and neural network has the worst. On window size 6 and 7 neural network has 50% accuracy that means it could not be used for detection on the proposed dataset. Boost has the second place with a bit higher error rate than Random Forest.

It is also noticeable that by increasing the window size the models would result in different characteristics such as Support Vector Machine. Accuracy of SVM has changed by 7 percentages from window size 5 (77.71%) to window size 8 (85.92). It is possible to get better results from this model on wider windows of sequences.

7. Future Works

It is possible to approach this problem from two different views. One would be the usage of one-class classifiers. With this kind of classifier we would just train the model by the normal data and anything other than the normal would be flagged as anomaly. This could solve the labeling challenge that we had for our proposed solution.

It is also possible to use the models that the window size is not an input. Models like Markov model and Hidden Markov model (HMM) could learn the whole sequence rather than subsets of the sequence. This could help us to see delayed-time anomalies that tried to be hidden from the ADS, by delaying the next system call to be able to hide the sequence.

8. ACKNOWLEDGMENTS

Our thanks to Wael Khreich for his useful advice and help us to find our dataset.

9. REFERENCES

[1] Creech, C. and Jiankun, H. A Semantic Approach to Host-based Intrusion Detection Systems Using Contiguous and Discontiguous System Call Patterns. IEEE Transactions on Computers. Vol. 99, 2013

[2] S. Forrest. A Sense of Self for UNIX Processes. In Proceedings of the IEEE Symposium on Security and Privacy, pages 120-128, Oakland, CA, May 1996.

[3] S. A., Forrest, S., and Somayaji, A. 1998. Intrusion detection using sequences of system calls. Journal of Computer Security 6, 3, 151–180.

[4] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. Advances in Neural Information Processing Systems, 9:155–161, 1997

[5] L. Breiman, Random Forests, Technical Report, January, 2001

[6] R. E. Schapire, The Boosting Approach to Machine Learning An Overview, Nonlinear Estimation and Classification, Springer, 2003

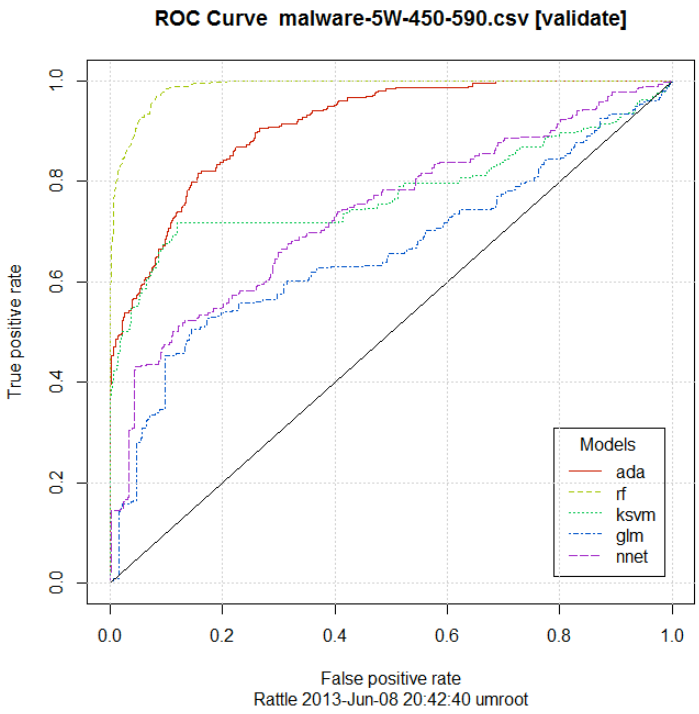
[7] David Wagner , Paolo Soto, Mimicry attacks on host-based intrusion detection systems, Proceedings of the 9th ACM conference on Computer and communications security, November 18-22, 2002, Washington, DC, USA

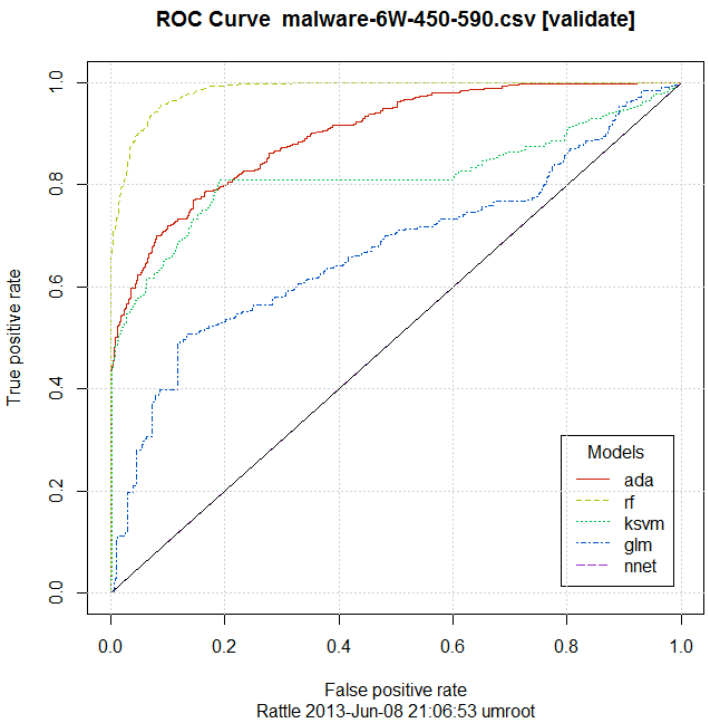
[8] C. Warrender, S. Forrest, B. Pearlmutter, Detecting Intrusions Using System Calls: Alternative Data Models

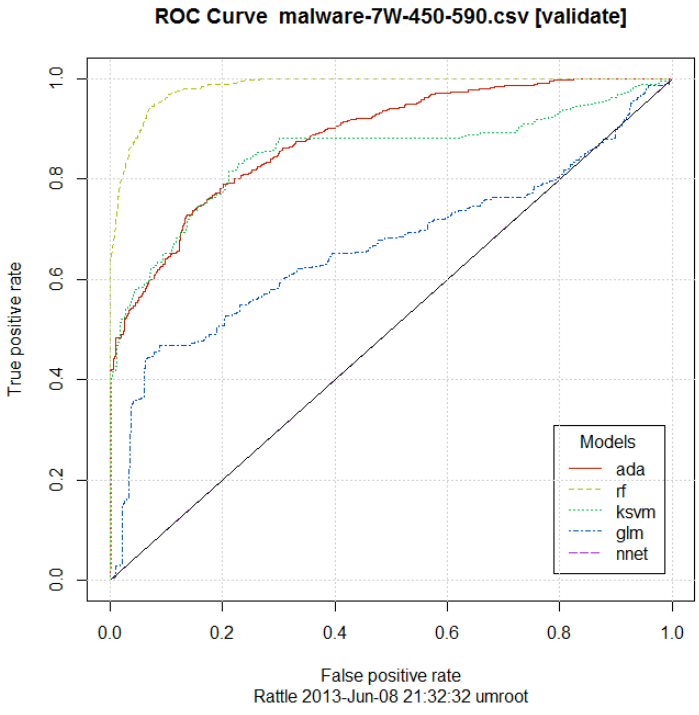
[9] V. Chandola, A. Banerjee, and V. Kumar, Anomaly detection: A survey. In ACM Comput. Surv., vol. 41, pp. 15:1-15:58, July 2009.

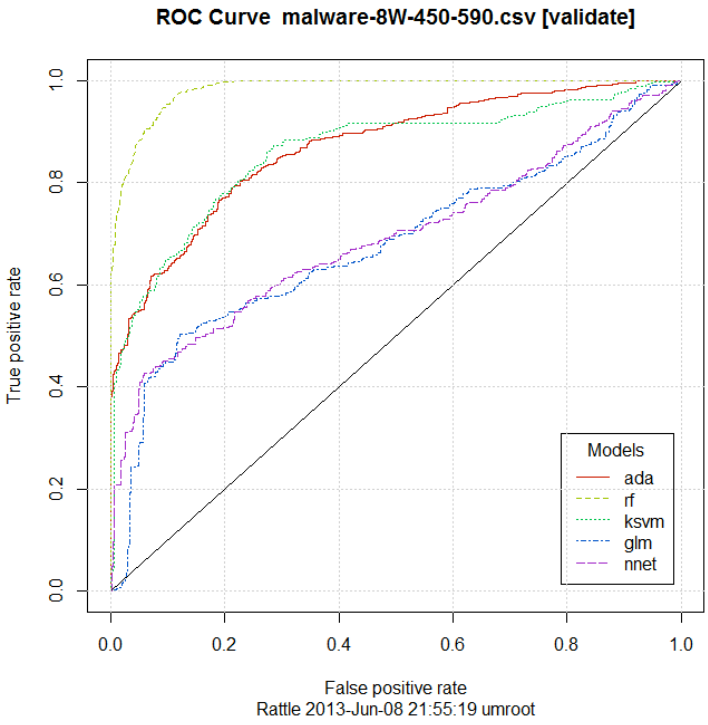
[10] S.A.Hofmeyr,S.Forrest,andA.Somayaji.Intrusiondetec- tion using sequences of system calls. Journal of Computer Security, 6(3), 1998.

[Appendix1]


ROC Curve malware-5W-450-590.csv [validate]
True positive rate
False positive rate
Rattle 2013-Jun-08 20:42:40 umroot
Models
ada
rf
ksvm
glm
nnet
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve malware-6W-450-590.csv [validate]
True positive rate
False positive rate
Rattle 2013-Jun-08 21:06:53 umroot
Models
ada
rf
ksvm
glm
nnet
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve malware-7W-450-590.csv [validate]
True positive rate
False positive rate
Rattle 2013-Jun-08 21:32:32 umroot
Models
ada
rf
ksvm
glm
nnet
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve malware-8W-450-590.csv [validate]
True positive rate
False positive rate
Rattle 2013-Jun-08 21:55:19 umroot
Models
ada
rf
ksvm
glm
nnet
0.0
0.2
0.4
0.6
0.8
1.0