10-2010

Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection

Proceso L. Fernandez Jr
Ateneo de Manila University, pfernandez@ateneo.edu

Jeffrey C. Yiu

Paul Albert R. Arana

Custom Citation

Yiu, J., Fernandez, P., & Arana, P. (2010). Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection. Philippine Information Technology Journal, 3(2), 26-32.

Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection

Jeffrey C. Yiu [*] Proceso L. Fernandez [†] Paul Albert R. Arana [‡]

ABSTRACT
Many malware detectors utilize data mining techniques as primary tools for pattern recognition. As the number of new and evolving malware continues to rise, there is an increasing need for faster and more accurate detectors. However, for a given malware detector, detection speed and accuracy are usually inversely related. This study explores several configurations of classification combined with feature selection. An optimization function involving accuracy and processing time is used to evaluate each configuration. A real data set provided by Trend Micro Philippines is used for the study. Among 18 different configurations studied, it is shown that J4.8 without feature selection is best for cases where accuracy is extremely important. On the other hand, when time performance is more crucial, applying a Naïve Bayes classifier on a reduced data set (using Gain Ratio Attribute Evaluation to select the top 35 features only) gives the best results.

Keywords
Malware Detection, Data Mining, Dimension Reduction, Feature Selection, Classification

1. INTRODUCTION
Malicious software, or malware, are harmful programs deliberately designed to compromise or damage a computer system's operations and data without the user's consent or knowing [2, 6, 9]. Malware include, but are not limited to, viruses, trojans, worms, and spyware [6, 9]. Due to the significant damage that malware can inflict [2, 6, 8], malware detectors have been developed to prevent malware from intruding or harming a system [2, 6].

Several malware detection techniques have been developed since the creation of the first virus. Commercial malware detectors mostly rely on the signature-based technique, where known malware patterns are used for classification [2, 10]. However, as malware continue to evolve and be produced at an increasing rate [5], signature bases have to be regularly updated in order to account for new malware features [2, 4, 10] which, as a result, tend to suffer from high dimensionality. In effect, detection performance is compromised [10].

In order to develop more efficient and effective detectors, the issue of high dimensionality has to be resolved. This can be achieved by applying dimension reduction techniques to signature bases, such that only the most relevant features are considered during classification. Theoretically, by averting the curse of dimensionality, a detector's performance should improve [1, 11, 13].

This paper is organized as follows. Section 2 briefly covers the literature most related to the study. A detailed description of our methodology is presented in Section 3. Results are presented and analyzed in the succeeding section. A summary of the main results and proposed future directions are covered in Section 4.

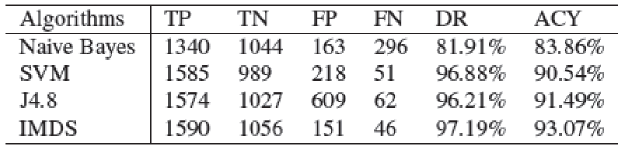
2. RELATED WORK
2.1 Comparison of Data Mining Techniques
The research conducted by [12] focused on the development of an Intelligent Malware Detection System (IMDS). Their experiment involved the comparison of several data mining techniques against IMDS. The classifiers used were Naïve Bayes, J4.8 decision tree, and Support Vector Machine (SVM). The data set was generated by randomly selecting 2,843 executables (1,207 benign; 1,636 malicious) and converting the signature base into a relational table, where each feature pertains to an API call. The feature set was then ranked, and only the top 500 features were selected for processing. Instead of separate training and testing sets, ten-fold cross validation was used to determine the accuracy of each classifier. Results showed that the IMDS has the highest accuracy, while Naïve Bayes has the lowest (See Figure 1). Although [12] has proven that the IMDS has superior detection rate compared to the other classifiers, the researchers failed to consider processing time in its evaluation.

[*]Jeffrey Yiu is an M.Sc. student of Computer Science in Ateneo de Manila University, Quezon City, Philippines.

[†]Proceso Fernandez is an Assistant Professor at the Ateneo de Manila University, Quezon City, Philippines

[‡]Paul Albert Arana is a member of the Antivirus Research and Development Group of Trend Micro Incorporated, Pasig City, Philippines

8th National Conference on I.T. Education (NCITE 2010)
La Carmela de Boracay Convention Center, Boracay, Aklan,
20 – 23 October 2010



Algorithms	TP	TN	FP	FN	DR	ACY
Naive Bayes	1340	1044	163	296	81.91%	83.86%
SVM	1585	989	218	51	96.88%	90.54%
J4.8	1574	1027	609	62	96.21%	91.49%
IMDS	1590	1056	151	46	97.19%	93.07%

Figure 1: Accuracy of different classifiers [12]

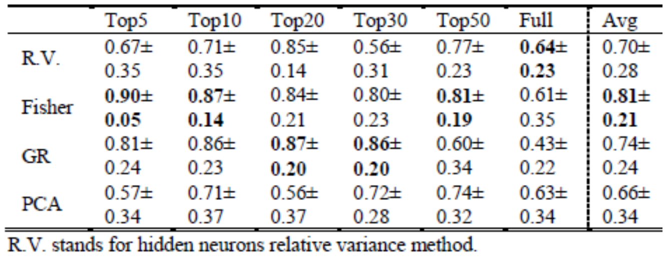
2.2 Comparison of Dimension Reduction Techniques

The research of [10], which deals with worm detection, affirms that applying feature selection techniques onto a data set may improve a classifier's detection rate. In the experiment, four different feature selection techniques (namely, Fisher's score ranking, Gain Ratio Filter, Principal Component Analysis, and a relevance variance ranker for neural networks) were applied onto a data set that contains a total of 323 features. Each selection technique was used to rank the entire feature set. Six feature subsets were produced by each accordingly: top 5, top 10, top 20, top 30, top 50, and full. A feed-forward neural network was used as the classifier. Results showed that the highest overall accuracy rate (0.90 ± 0.05) was produced by using the top 5 features identified by Fisher's score technique. Accuracy in this context is computed using the formula

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

that involves counting the True Positves (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

It was concluded that the benefits of feature selection, in terms of accuracy, only applies if the most important features are used for processing. However, without using another classifier, the statement may only be true for neural networks, or learning algorithms in general. The conclusion may not hold if other classifiers were to be run on the reduced data sets as well.



	Top5	Top10	Top20	Top30	Top50	Full	Avg
R.V.	0.67± 0.35	0.71± 0.35	0.85± 0.14	0.56± 0.31	0.77± 0.23	**0.64± 0.23**	0.70± 0.28
Fisher	**0.90± 0.05**	**0.87± 0.14**	0.84± 0.21	0.80± 0.23	**0.81± 0.19**	0.61± 0.35	**0.81± 0.21**
GR	0.81± 0.24	0.86± 0.23	**0.87± 0.20**	**0.86± 0.20**	0.60± 0.34	0.43± 0.22	0.74± 0.24
PCA	0.57± 0.34	0.71± 0.37	0.56± 0.37	0.72± 0.28	0.74± 0.32	0.63± 0.34	0.66± 0.34

R.V. stands for hidden neurons relative variance method.

Figure 2: Worm classification with feature selection [10]

2.3 Comparison of Data Mining Techniques on Reduced Data Sets

The study conducted by [11] compared the performances of intrusion detection methods on reduced data sets. The research used the KDDCup '99 data set, which contains 41 features and 5 classes. The selection techniques used were Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA). Each technique reduced the size of the original feature set to 13, 12, and 17 features respectively. Six classifiers were run on each of the reduced set, namely Gaussian Mixture, Basis Function, SOM, Binary Tree, ART, and LAMSTAR. Results showed that the feature set selected using PCA improved the overall detection rate of the classifiers, among which the LAMSTAR method produced the highest. With regards to processing time, the differences in training and testing times with the other classifiers were not significant. The research, however, did not explicitly provide a baseline result set, where the classifiers are run on the original data set; hence, differences in the classifiers' performances on a non-reduced data set versus a reduced one cannot be concretely established.

3. METHODOLOGY

3.1 Data Acquisition

The data set for the experiment of this research was provided by Trend Micro Philippines, Inc., which specializes in malware detection and cleaning. Third-party file information dumping tools were used to collect both static and dynamic file features. Static features are those obtained by simply investigating the binary contents of a file. Dynamic features require executing the file in a sandbox-type of monitoring system.

The list of tools used for static feature extraction are given below:

- PE_Dump - used to dump the PE file features of the file
- Bintext - used to dump the readable ANSI and UNICODE text strings in the file
- Trend Micro VSDT tool - extracts the resource information from the file
- Trend Micro ScanPacker tool - extracts the packer information from the file
- Trend Micro Entropy tool - extracts the entropy computations on the file sections

The type of information obtained by the tools are as follows:

- Windows Libraries and Functions Used
- Extracted Significant Readable Text Strings
- File Structure Details (Portable Executable File header, section tables, resource table, Import/Export tables)
- Compression Details (Entropy, Packer Type/Layer)
- Malware-related URL's in the code

For dynamic file features, the samples were executed by an automated system using the Trend Micro sandbox technology. The information gathered here are as follows:

- File Events
- Process/Services Events
- Registry Events
- Network Events

The rule set utilizes two types of feature matching:

- RULE - matches a single feature with a file sample record in the database (e.g. filesize <1000 bytes)
- LOGIC - matches a conjunction of RULES (e.g. RULE01 & RULE02 & RULE 03) with a file sample

Each of the rules represents a particular feature being checked. Each LOGIC field signifies a RULE combination.

The resulting data set file contains a table that profiles malware instances. This involves 60,107 rows that refer to the file collection, and 143 feature values that were collected as described previously. Each entry has been classified either as malicious or benign. Of the entire set, 52,881 entries are benign while 7,226 are malicious (approx. 12%).

3.2 Data Processing

This study used Weka's Java library [3]. A separate Java application was developed for the combined feature selection with classification since the Weka GUI does not provide batch filtering and standardization functions necessary for running test sets on trained classifiers.

Due to the large size of the data set, several classfiers and feature selection tools could not be executed. Thus, the study focuses on two classifiers and several filters. The classifiers used in the experiment are (a) Naïve Bayes and (b) J4.8 decision tree. The feature selection techniques used are as follows (attribute evaluator with search method): (a) Correlation-based Feature Subset Selection (CFS Subset Evaluation) with Best-First search; (b) Filtered Subset Evaluation with Greedy Stepwise search; (c) Chi Squared Attribute Evaluation with Ranker; and (d) Gain Ratio Attribute Evaluation with Ranker. Both classifiers and all selection techniques used Weka's default settings.

Classification was first conducted on the non-reduced set to provide a baseline for comparison purposes. Afterwards, a feature selection technique was applied onto the data set, and then the two classifiers were trained and tested on the resulting reduced set. Similar steps were applied for each of the other reduction techniques. For each configuration, a tenfold cross-validation was performed in order to compute the accuracy of malware detection.

3.3 Data Analysis

For this research, a classifier's performance is computed using an optimization function that involves both the accuracy and processing time.

$$OPT(C) = \alpha \cdot A(C) + (1 - \alpha) \cdot T(C)$$

where $A(C)$ and $T(C)$ are the normalized accuracy and normalized processing time, respectively, of the classifier C, and $\alpha \in [0, 1]$ is a parameter used to indicate the relative weights of these two values. In particular, $\alpha > 0.5$ indicates that accuracy is prioritized over processing time. Similarly, $\alpha < 0.5$ indicates that the processing time is more preferred. On the extreme cases, the optimization function is dependent only on the accuracy if $\alpha = 1$, or on the processing time if $\alpha = 0$.

Since $A(C)$ and $T(C)$ are normalized values, the value of $OPT(C)$ also ranges from 0 to 1, with $OPT(C) = 1$ being the ideal case.

The accuracy ACY is normalized using

$$A(C) = \frac{ACY_C - ACY_{min}}{ACY_{max} - ACY_{min}}$$

For the normalized time, the raw time T (in milliseconds) is first computed using $T = t_1 + t_2 + t_3$, where
t_1 = feature selection duration
t_2 = training duration
t_3 = testing duration

Then the normalized time $T(C)$ of the classifier C is computed using

$$T(C) = 1 - \frac{T_C - T_{min}}{T_{max} - T_{min}}$$

Note that a score of 1 is given to the fastest classifier (i.e., the processing time is minimum) while a score of 0 is given to the slowest (maximum processing time) classifier.

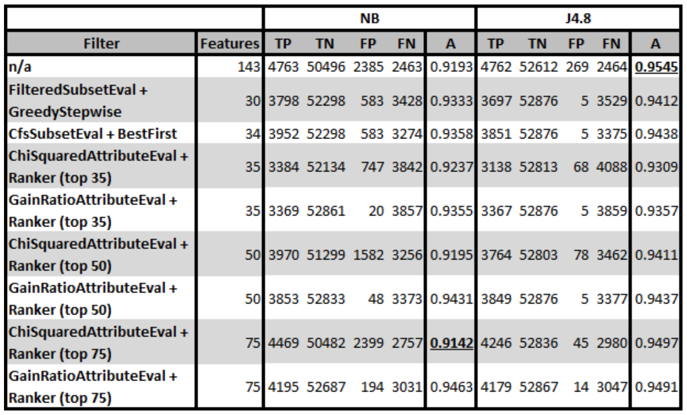
For each classifier, the accuracy and processing time are recorded. The optimization function $OPT(C)$ is then computed several times, using different α values. A higher resulting $OPT(C)$ value indicates a better overall performance for the classifier.

3.4 Test Environment

The experiments were run on a machine with the following specifications: (OS) Microsoft Windows 7 32-bit; (Processor) Intel Pentium dual-core T4200 (2.0 Ghz, 8mm Mhz FSB, 1 MB L2 cache); (RAM) 4Gb DDR2 @ 332 Mhz; (MoBo) Acer Aspire 4730Z. The system's activity was monitored to ensure that the test environment for all instances of the experiment would be similar to each other. Memory usage averaged at 27%, while CPU usage averaged at 15%.

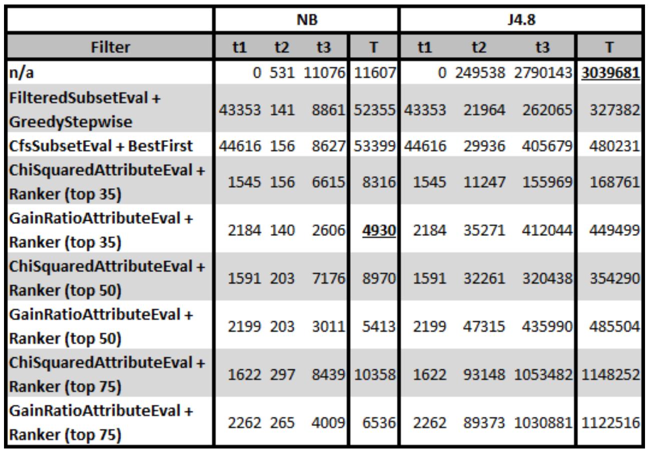
4. RESULTS AND ANALYSIS

After running one experiment for each classifier-reduction combination, the results are as follows:



		NB					J4.8				
Filter	Features	TP	TN	FP	FN	A	TP	TN	FP	FN	A
n/a	143	4763	50496	2385	2463	0.9193	4762	52612	269	2464	**0.9545**
FilteredSubsetEval + GreedyStepwise	30	3798	52298	583	3428	0.9333	3697	52876	5	3529	0.9412
CfsSubsetEval + BestFirst	34	3952	52298	583	3274	0.9358	3851	52876	5	3375	0.9438
ChiSquaredAttributeEval + Ranker (top 35)	35	3384	52134	747	3842	0.9237	3138	52813	68	4088	0.9309
GainRatioAttributeEval + Ranker (top 35)	35	3369	52861	20	3857	0.9355	3367	52876	5	3859	0.9357
ChiSquaredAttributeEval + Ranker (top 50)	50	3970	51299	1582	3256	0.9195	3764	52803	78	3462	0.9411
GainRatioAttributeEval + Ranker (top 50)	50	3853	52833	48	3373	0.9431	3849	52876	5	3377	0.9437
ChiSquaredAttributeEval + Ranker (top 75)	75	4469	50482	2399	2757	**0.9142**	4246	52836	45	2980	0.9497
GainRatioAttributeEval + Ranker (top 75)	75	4195	52687	194	3031	0.9463	4179	52867	14	3047	0.9491

Figure 3: Accuracy per classifier and reduced data set



	NB				J4.8			
Filter	**t1**	**t2**	**t3**	**T**	**t1**	**t2**	**t3**	**T**
n/a	0	531	11076	11607	0	249538	2790143	**<u>3039681</u>**
FilteredSubsetEval + GreedyStepwise	43353	141	8861	52355	43353	21964	262065	327382
CfsSubsetEval + BestFirst	44616	156	8627	53399	44616	29936	405679	480231
ChiSquaredAttributeEval + Ranker (top 35)	1545	156	6615	8316	1545	11247	155969	168761
GainRatioAttributeEval + Ranker (top 35)	2184	140	2606	**<u>4930</u>**	2184	35271	412044	449499
ChiSquaredAttributeEval + Ranker (top 50)	1591	203	7176	8970	1591	32261	320438	354290
GainRatioAttributeEval + Ranker (top 50)	2199	203	3011	5413	2199	47315	435990	485504
ChiSquaredAttributeEval + Ranker (top 75)	1622	297	8439	10358	1622	93148	1053482	1148252
GainRatioAttributeEval + Ranker (top 75)	2262	265	4009	6536	2262	89373	1030881	1122516

Figure 4: Processing time per classifier and reduced data set

Figures 3 and 4 show the raw accuracy and processing time for each classifier-reduction combination, while Figures 5 onwards show the normalized accuracy and time, as well as the optimization values, per combination.

For brevity, the classifiers and reduction techniques would be represented from this point onwards as follows: Naïve Bayes (NB); CFS Subset Evaluation (CFS); Filtered Subset Evaluation (FSE); Chi Squared (CS); and Gain Ratio (GR). The latter two would further be associated with a number that corresponds to the number of features selected (i.e. CS 35 for Chi Squared selecting the top 35 features). Combinations would be represented in the form of $\textit{(classifier)}_{(reduction)}$.

4.1 Accuracy Analysis

Based on the results, the classifier-reduction combination with the highest accuracy is under J4.8 without using feature selection (95.45%), while the lowest accuracy is under NB_{CS75} (91.42%).

It is observed that NB tends to prefer GR among the given feature selection techniques. As the number of features increased, so did NB's accuracy under GR, while, conversely, it deteriorated under CS. In fact, the accuracy of NB without feature selection is higher than its accuracy under CS 75. Incidentally, the latter also has the lowest accuracy across all 18 classifier-reduction combinations. Conversely, NB's highest accuracy is under GR 75. It should be noted that, although GR rearranges the feature set from most relevant to least relevant, NB uses joint probability distribution, and therefore the order of features does not matter; hence, in the case of NB, it is safe to assume that GR provides more relevant feature sets than the rest of the given reduction techniques. This also affirms the conclusion of [10], which states that the benefits of feature selection only applies when the most important features are chosen. Despite both having selected the same numbers of features, the feature sets provided by CS are apparently inferior to those by GR.

As indicated earlier, J4.8 has the highest accuracy across all 18 classifier-reduction combinations, though surprisingly under the non-reduced data set. In fact, results showed that the accuracy of J4.8 increases proportionally to the number of features, which appears to disprove the supposed negative effects of high dimensionality, at least from this perspective. Additionally, unlike NB, J4.8 does not appear to prefer any particular reduction technique, such that its accuracy increases with the number of features for both CS and GR, though the latter still has a higher accuracy than the former, albeit insignificantly.

The results also validate those in [7] which mentions that NB is highly sensitive to redundant features while decision trees are not. The data set, which inherently has redundant features, places NB at a disadvantage. The disparity in accuracy between NB and J4.8 may have been caused by such. This is evidenced by the results in the non-reduced set, where J4.8 has a significant lead of approximately 3.5% over NB. As results showed, applying feature selection generally improves the accuracy of NB. It was also revealed that the large difference in accuracy mostly lies in the False Positive (FP) detection rate, where NB tends to misclassify more than J4.8, regardless whether the data set is reduced or not. Again, NB appears to misclassify the least when using the data set filtered with GR.

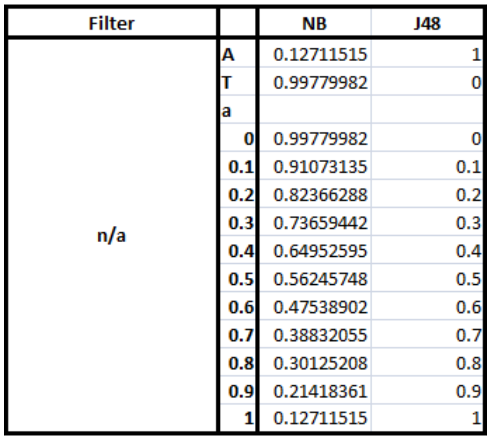
Practicality-wise, FP's or false alarms are not as dangerous as False Negatives (FN) since it is in the latter case where malware are able to intrude a system. Surprisingly, both NB's and J4.8's FN rates are statistically close for all classifier-reduction combinations.

4.2 Runtime Analysis

Results showed that NB consistently has a significantly lower processing time (T) than J4.8. The fastest processing time is under NB_{GR35} (4,930ms), while the longest duration is under J4.8 using the non-reduced set (3,039,681ms).

The processing times of NB, with or without feature selection, generally do not differ much from each other. In fact, the feature selection time (t_1) of CFS and FSE even made it siginificantly longer. Excluding t_1, however, improves NB's processing time by at least 22%. It is also observed that the growth rate of NB's processing time is linearly proportional to the size of the given feature set.

J4.8, on the other hand, significantly benefits from feature selection. Even with the large overhead time of CFS and FSE, the total processing time of J4.8 still improved by as much as 84%. The fastest processing time of J4.8 is under CS 35 (168,761ms). As expected, the processing time of J4.8 increases rapidly as the number of features increases. The results of CS and GR seem to indicate that the processing time of J4.8 has an exponential growth rate with respect to the number of features. However, this might only hold true for ranking filters. It was observed that CS 35 is much faster than CFS and FSE in all aspects despite the latter two having only selected 34 and 30 features respectively. It was further observed that J4.8 appears to have an initial bias for CS over GR, though eventually J4.8_{GR} would be faster than J4.8_{CS}. This suggests that the processing time of J4.8 is more strongly influenced by the feature set itself rather than its size. Again, this confirms the conclusion of [10], though from a time perspective.



Filter		NB	J48
n/a	A	0.12711515	1
	T	0.99779982	0
	a		
	0	0.99779982	0
	0.1	0.91073135	0.1
	0.2	0.82366288	0.2
	0.3	0.73659442	0.3
	0.4	0.64952595	0.4
	0.5	0.56245748	0.5
	0.6	0.47538902	0.6
	0.7	0.38832055	0.7
	0.8	0.30125208	0.8
	0.9	0.21418361	0.9
	1	0.12711515	1

Figure 5: Baseline results

4.3 Optimization Analysis

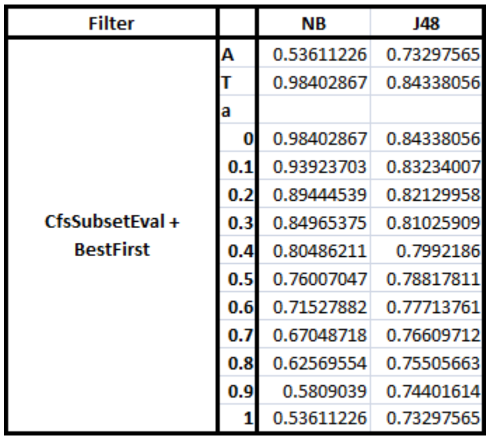
For cases where both accuracy and speed are equally prioritized, the best classifier-reduction combination can be identified by comparing OPT(C|α=0.5) across all combinations. Given such, the most optimal combination is under NB$_{GR75}$ (0.8983), while the least is under NB$_{CS75}$ (0.4991). For J4.8, the most optimal combination is under CFS (0.7882), while the least is under the non-reduced data set (0.5). It should be noted that equal priority in accuracy and speed is not similar to the concept of economy or "efficiency", as in accuracy per processing time or vice-versa; rather, it implies the average-case performance of a given combination.

The slope of a combination's graph indicates its orientation for either accuracy or speed, where a rising graph indicates that the combination would tend to provide better accuracy performance than time, while a declining graph indicates otherwise. Figure 10 shows that all the graphs under NB have declining slopes. This indicates that, regardless of classifier-reduction combination, NB would tend to have better time performance than accuracy. J4.8, on the other hand, does not have a uniform orientation. Figure 11 reveals that the slopes of the graphs appear to be proportional to the number of features, such that the graphs rise as the feature size increases. Aside from the baseline, only the graphs of J4.8$_{CS75}$ and J4.8$_{GR75}$ have a positive slope.

The computed optimization values also allows for the identification of combinations with equal performances given a certain priority. For NB, Figure 10 shows that most graphs intersect roughly at $\alpha = 0$. This means that, if time is the only priority, then most of the combinations under NB would give approximately equivalent performance levels. Figure 12 shows that NB$_{GR35}$ and J4.8$_{GR35}$ are equivalent roughly at α = 1. Since NB processes significantly faster than J4.8, and that J4.8$_{GR35}$ only has a 0.02% better accuracy, NB$_{GR35}$ might be the more practical choice between the two given priority in accuracy.

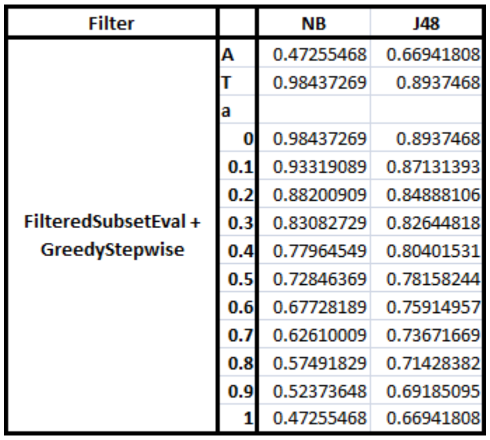
5. CONCLUSION

In this study, several combinations of feature selection strategies combined with standard classifiers were applied for malware detection. An optimization function involving the ac-



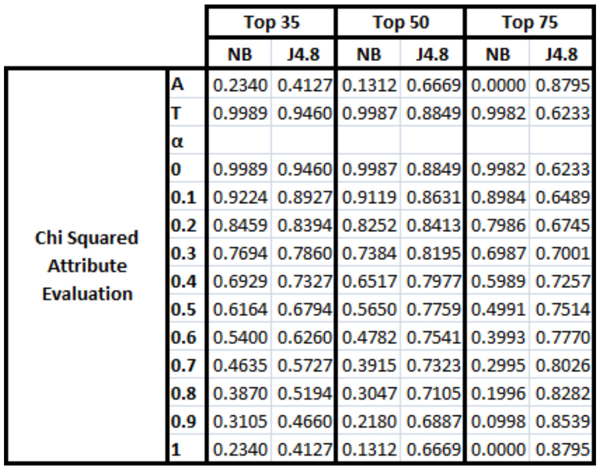
Filter		NB	J48
CfsSubsetEval + BestFirst	A	0.53611226	0.73297565
	T	0.98402867	0.84338056
	a		
	0	0.98402867	0.84338056
	0.1	0.93923703	0.83234007
	0.2	0.89444539	0.82129958
	0.3	0.84965375	0.81025909
	0.4	0.80486211	0.7992186
	0.5	0.76007047	0.78817811
	0.6	0.71527882	0.77713761
	0.7	0.67048718	0.76609712
	0.8	0.62569554	0.75505663
	0.9	0.5809039	0.74401614
	1	0.53611226	0.73297565

Figure 6: CFS Subset Evaluation with Best First search



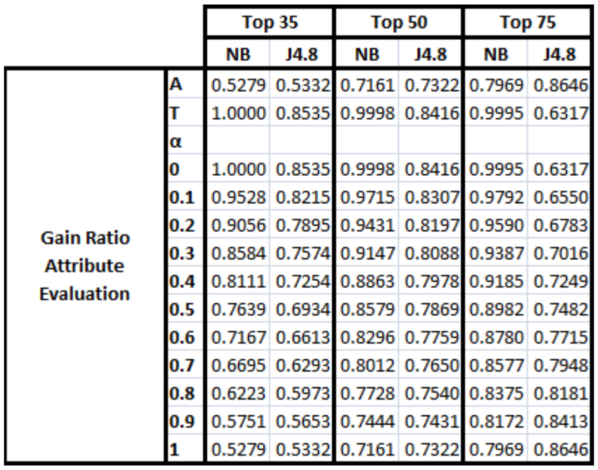
Filter		NB	J48
FilteredSubsetEval + GreedyStepwise	A	0.47255468	0.66941808
	T	0.98437269	0.8937468
	a		
	0	0.98437269	0.8937468
	0.1	0.93319089	0.87131393
	0.2	0.88200909	0.84888106
	0.3	0.83082729	0.82644818
	0.4	0.77964549	0.80401531
	0.5	0.72846369	0.78158244
	0.6	0.67728189	0.75914957
	0.7	0.62610009	0.73671669
	0.8	0.57491829	0.71428382
	0.9	0.52373648	0.69185095
	1	0.47255468	0.66941808

Figure 7: Filtered Subset Evaluation with Greedy Stepwise search



		Top 35		Top 50		Top 75	
		NB	J4.8	NB	J4.8	NB	J4.8
Chi Squared Attribute Evaluation	A	0.2340	0.4127	0.1312	0.6669	0.0000	0.8795
	T	0.9989	0.9460	0.9987	0.8849	0.9982	0.6233
	α						
	0	0.9989	0.9460	0.9987	0.8849	0.9982	0.6233
	0.1	0.9224	0.8927	0.9119	0.8631	0.8984	0.6489
	0.2	0.8459	0.8394	0.8252	0.8413	0.7986	0.6745
	0.3	0.7694	0.7860	0.7384	0.8195	0.6987	0.7001
	0.4	0.6929	0.7327	0.6517	0.7977	0.5989	0.7257
	0.5	0.6164	0.6794	0.5650	0.7759	0.4991	0.7514
	0.6	0.5400	0.6260	0.4782	0.7541	0.3993	0.7770
	0.7	0.4635	0.5727	0.3915	0.7323	0.2995	0.8026
	0.8	0.3870	0.5194	0.3047	0.7105	0.1996	0.8282
	0.9	0.3105	0.4660	0.2180	0.6887	0.0998	0.8539
	1	0.2340	0.4127	0.1312	0.6669	0.0000	0.8795

Figure 8: Chi Squared Optimization Values



		Top 35		Top 50		Top 75	
		NB	J4.8	NB	J4.8	NB	J4.8
Gain Ratio Attribute Evaluation	A	0.5279	0.5332	0.7161	0.7322	0.7969	0.8646
	T	1.0000	0.8535	0.9998	0.8416	0.9995	0.6317
	α						
	0	1.0000	0.8535	0.9998	0.8416	0.9995	0.6317
	0.1	0.9528	0.8215	0.9715	0.8307	0.9792	0.6550
	0.2	0.9056	0.7895	0.9431	0.8197	0.9590	0.6783
	0.3	0.8584	0.7574	0.9147	0.8088	0.9387	0.7016
	0.4	0.8111	0.7254	0.8863	0.7978	0.9185	0.7249
	0.5	0.7639	0.6934	0.8579	0.7869	0.8982	0.7482
	0.6	0.7167	0.6613	0.8296	0.7759	0.8780	0.7715
	0.7	0.6695	0.6293	0.8012	0.7650	0.8577	0.7948
	0.8	0.6223	0.5973	0.7728	0.7540	0.8375	0.8181
	0.9	0.5751	0.5653	0.7444	0.7431	0.8172	0.8413
	1	0.5279	0.5332	0.7161	0.7322	0.7969	0.8646

Figure 9: Gain Ratio Optimization Values

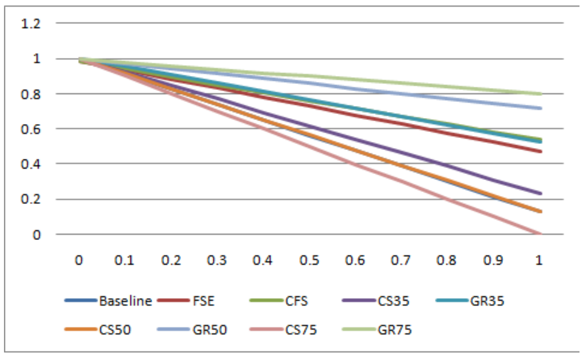


Figure 10: Naïve Bayes Opt. Values (Graph)

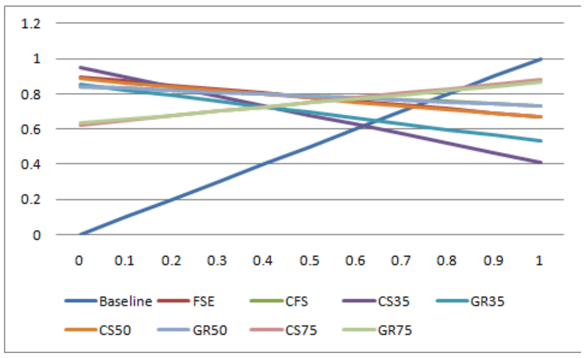


Figure 11: J4.8 Optimization Values (Graph)

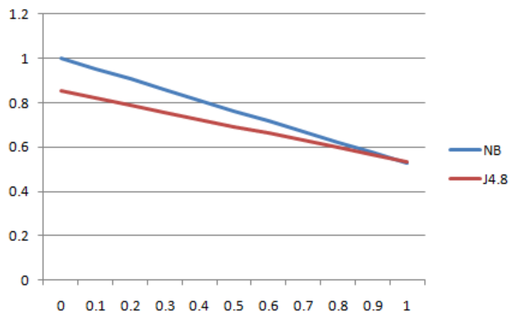


Figure 12: Gain Ratio (Top 35) Opt. Values (Graph)

curacy and processing time was used for comparing the different configurations. The data set used contains 60,107 files (approximately 12% are infected with malware) from which 143 features were recorded.

It is shown that when accuracy is extremely important, J4.8 without any feature selection works best. It returned correct classification 95.45% of the time after 3,039 seconds (or 50.65 minutes) of processing.

On the other hand, when time performance is a lot more crucial, applying a Naïve Bayes classifier on a reduced data set (using Gain Ratio Attribute Evaluation to select the top 35 features only) gives the best results. Within 4.93 seconds, it is able to finish processing the data set and return with a respectable 94.31% accuracy.

When both the accuracy and processing time are given equal importance, then Naive Bayes combined with Gain Ratio Attribute Evaluation, selecting the top 75 features, gives the most impressive results. After 6.55 seconds, it is able to classify the files in the data set with 94.63% accuracy.

Future studies may explore other classifiers and filters as there are many possible classifier and reduction combinations. It would be interesting to find out which combination will further improve the results gathered in this study.

A different type of data set may also be used for data mining. Many data sets, in fact, are available online for different problem domains. A comparative analysis of classifiers and filters on different data sets might provide more insights on which classifiers and filters work best on which types of data sets.

6. REFERENCES

[1] S. Aksoy. Feature reduction and selection. Bilkent University, 2008.

[2] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. Semantics-aware malware detection. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 32–46. IEEE Computer Society, 2005.

[3] http://www.cs.waikato.ac.nz/ml/weka/.

[4] J. Z. Kolter and M. A. Maloof. Learning to detect malicious executables in the wild. In *KDD '04:*

Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 470–478, New York, NY, USA, 2004. ACM.

[5] T. Micro. 2008 annual threat roundup and 2009 forecast, 2009.

[6] V. P., V. Laxmi, and M. Gaur. Survey on malware detection methods. In *3rd Hackers' Workshop on Computer and Internet Security*, pages 74–79. Indian Institute of Technology Kanpur, 2009.

[7] C. A. Ratanamahatana and D. Gunopulos. Scaling up the naive bayesian classifier: Using decision trees for feature selection. In *Proceedings of Workshop on Data Cleaning and Preprocessing (DCAP 2002), at IEEE International Conference on Data Mining (ICDM 2002)*, 2002.

[8] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 38, Washington, DC, USA, 2001. IEEE Computer Society.

[9] M. A. Siddiqui. *Data Mining Methods for Malware Detection*. PhD thesis, University of Central Florida, 2008.

[10] D. Stopel, Z. Boger, R. Moskovitch, Y. Shahar, and Y. Elovici. Improving worm detection with artificial neural networks through feature selection and temporal analysis techniques. *International Journal of Applied Mathematics and Computer Sciences*, 1(1):34–40, 2005.

[11] V. Venkatachalam and S. Selvan. Performance comparison of intrusion detection system classifiers using various feature reduction techniques. *International Journal of Simulation Systems, Science and Technology*, 9(1):30–39, February 2008.

[12] Y. Ye, D.-D. Wang, T. Li, and D. Ye. IMDS: Intelligent malware detection system. In *International Conference on Knowledge Discovery and Data Mining*, pages 1043–1047. ACM, 2007.

[13] L. Yu, J. Ye, and H. Liu. Dimensionality reduction for data mining- techniques, applications and trends, 2008. http://www.cs.binghamton.edu/ lyu/SDM07/DR-SDM07.pdf.